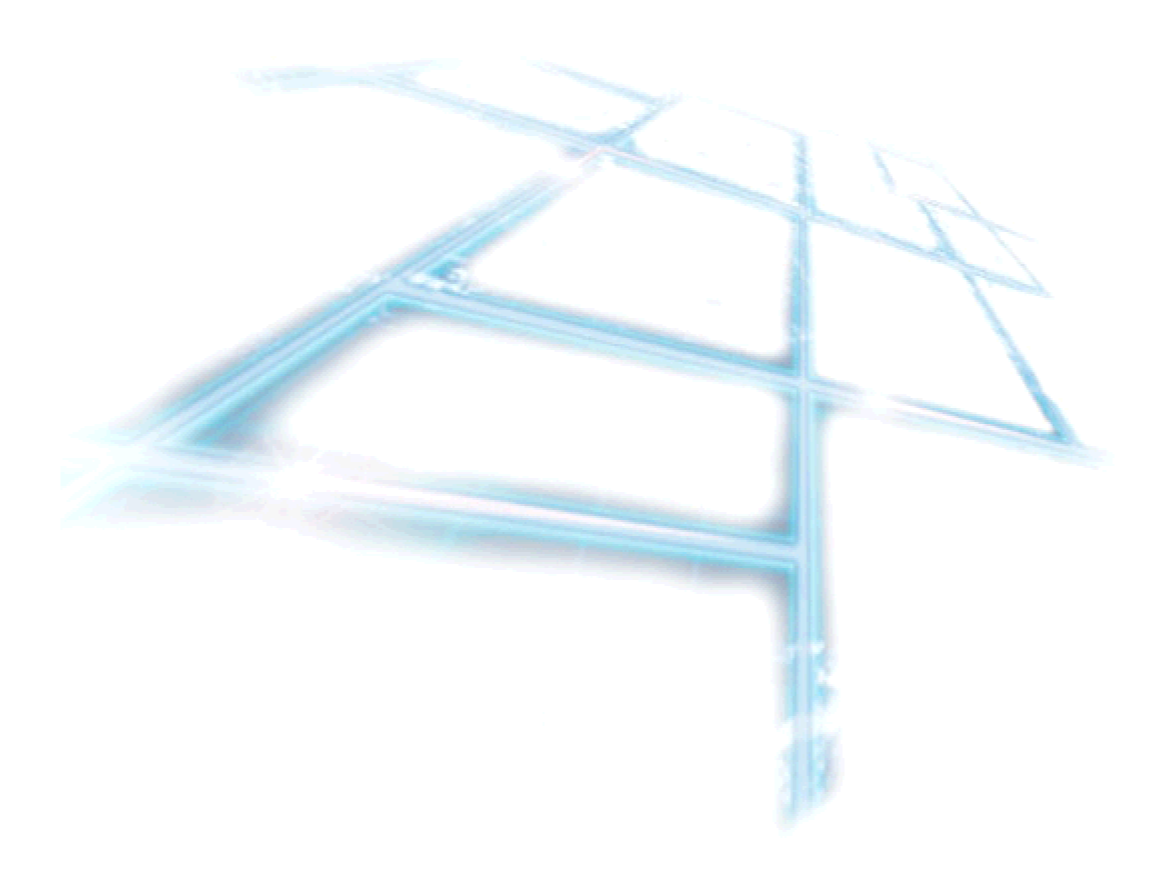


USB-GPIB-1 / PCIGPIB-1 / GPIB-1





Wesseling, November 2011 Version 4.1.4

Inhaltsverzeichnis

QUANCOM INFORMATIONSSYSTEME

1 Überblick

1.1 Einführung

Herzlichen Glückwunsch zum Kauf Ihrer QUANCOM Meßtechnik Karte. Sie sind nun stolzer Besitzer einer modernen Meß- Steuer- Regelkarte, die den neusten Stand der Technik darstellt und deren Eigenschaften und Funktionalität sich mit jedem anderem Gerät dieser Klasse messen kann und sie sogar noch übertrifft. Zu den besonderen Eigenschaften dieser Karte gehören:

Eigenschaften der Karte

- einfach programmierbar
- diverse Beispielprogramme in verschiedenen Programmiersprachen

1.2 Unsere Erfahrung ist ihr Gewinn

Wir von QUANCOM sind auf die Entwicklung für Hardware und Software spezialisiert. QUANCOM gehört mittlerweile zu einem der führenden Lieferanten für Meßtechnik und Automatisierung. In ihrem Entwicklungszentrum hat QUANCOM eine eindrucksvolle Produktpalette entwickelt.

1.3 Kommunikation mit unseren Kunden

QUANCOM möchte gerne Ihren Kommentar zu unseren Produkten und zu unseren Handbüchern. Wir sind an Ihren Anwendungen interessiert, die Sie mit unseren Produkten entwickeln. Wir möchten gleichzeitig helfen, wenn Sie Probleme haben. Um das zu vereinfachen enthält dieses Handbuch Kommentar- und Konfigurationsformulare, mit denen man direkt mit uns in Verbindung treten kann. Diese Formulare befinden sich in dem Kapitel "**Dokumentations Formular**" am Ende dieses Handbuches.

1.4 Änderungen zu diesem Handbuch und Softwareupdates

QUANCOM - Produkte zeichnen sich u.a. durch stetige Weiterentwicklung aus. Aktuelle Informationen über Änderungen können Sie den README - Dateien auf der Installations-CD entnehmen. Weitere Informationen und kostenlose Softwareupdates können Sie jederzeit auf den QUANCOM Internet – WWW-Seiten unter **www.quancom.de** erhalten.

1.5 Lieferumfang

• GPIB-Interface (Karte/Modul)
• Benutzerhandbuch (auf der CD als PDF)
• QUANCOM CD
• USB-Kabel (nur USB Version)

Sollten eine oder mehrere Komponenten fehlen wenden Sie sich bitte an Ihren Händler. QUANCOM behält sich das Recht vor, Änderungen im Lieferumfang ohne Vorankündigung vorzunehmen.

2 Technische Hardware Beschreibung

2.1 Allgemeines

Beim IEEE488-Bussystem liegen alle Instrumente parallel zu den Kommunikationsleitungen. Das Bussystem besteht aus 16 Signalleitungen: Acht Datenleitungen, drei ‚handshake'-Leitungen (data-byte-transfer-control) und fünf Leitungen zur Organisation zwischen den Instrumenten.

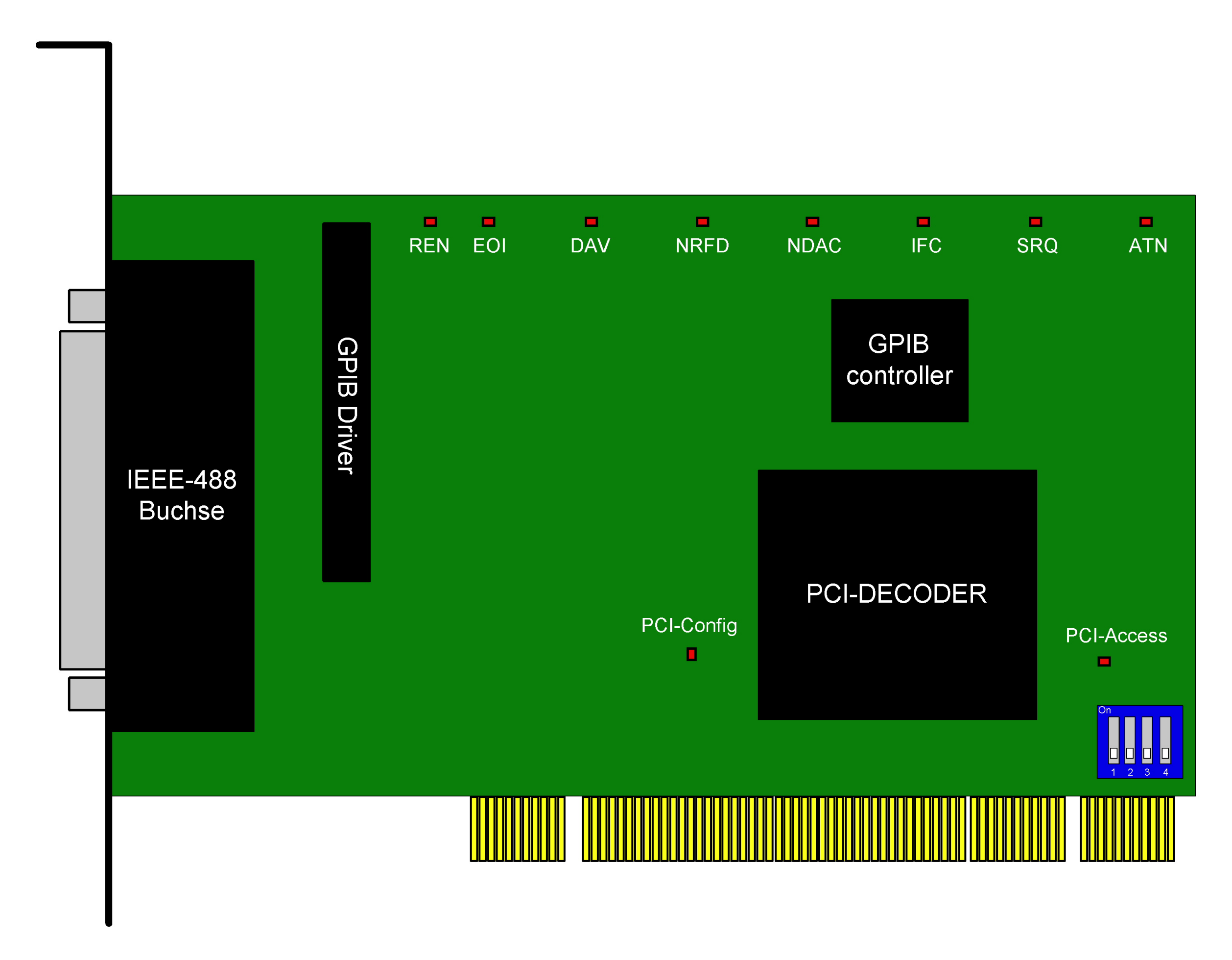
Ein Instrument wird als Sender (‚talker') und eines oder mehrere andere werden als Empfänger (‚listener') definiert. Diese Aufgaben übernimmt die GPIB-Controllerkarte.

2.2 Funktionsweise der GPIB-Karte

Die GPIB-Karte bietet über die Datenleitungen jedem Instrument eine bestimmte Adresse an. Diese Adresse steht zeitgleich an allen Instrumenten. Jedes Instrument vergleicht nun seine eigene (eingestellte) Adresse mit der Anliegenden. Stimmt sie überein, so wird das Gerät initialisiert um Befehle aufzunehmen. Anschließend gibt die Controllerkarte an die initialisierten Instrumente den Befehl, als ‚talker' oder ‚listener' zu arbeiten. Sind alle Geräte initialisiert, die am Datenaustausch beteiligt sein sollen, wird mittels der Dreidraht-handshake-funktion der Datentransfer organisiert und überwacht.

2.3 PCIGPIB (Rev. 4.xx)

2.3.1 Kartenübersicht PCIGPIB-1



2.3.2 Technische Daten

Controller:	µPD7210 kompatibel (mit HW FIFO)
Steckverbinder:	24 pol. IEEE-488 Buchse mit M3.5 Gewinde.
Leuchtdioden:	8 LEDs befinden sich auf der Karte für die Anzeige der Steuerleitungen ATN, SRQ, NDAC, NRFD, DAV, IFC, REN, EOI. Sowie 2 LEDs um einen Zugriff auf die Karte und den PCI-Konfigurations Status anzuzeigen.
Software:	Beispielsoftware in C/C++, Visual Basic, Lotus NotesLabWindows und Delphi

2.3.3 Einstellung der Moduladresse mit dem Dip-Schalter

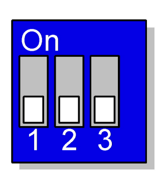
SW1	SW2	SW3	Moduladresse
Off	Off	Off	0 (default)
On	Off	Off	1
Off	On	Off	2
On	On	Off	3
Off	Off	On	4
On	Off	On	5
Off	On	On	6
On	On	On	7

Diese Tabelle zeigt, wie man mit Hilfe der Dipschalter die Moduladresse des Gerätes verändern kann.

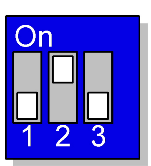
In den darunter liegenden Beispielen sind die Adressen 0 und 2 eingestellt.

Beispiele:

Adresse 0



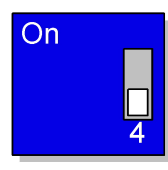
Adresse 2



2.3.4 Wahl zwischen dem I/O- und dem Memory-Modus mit Dipschalter 4

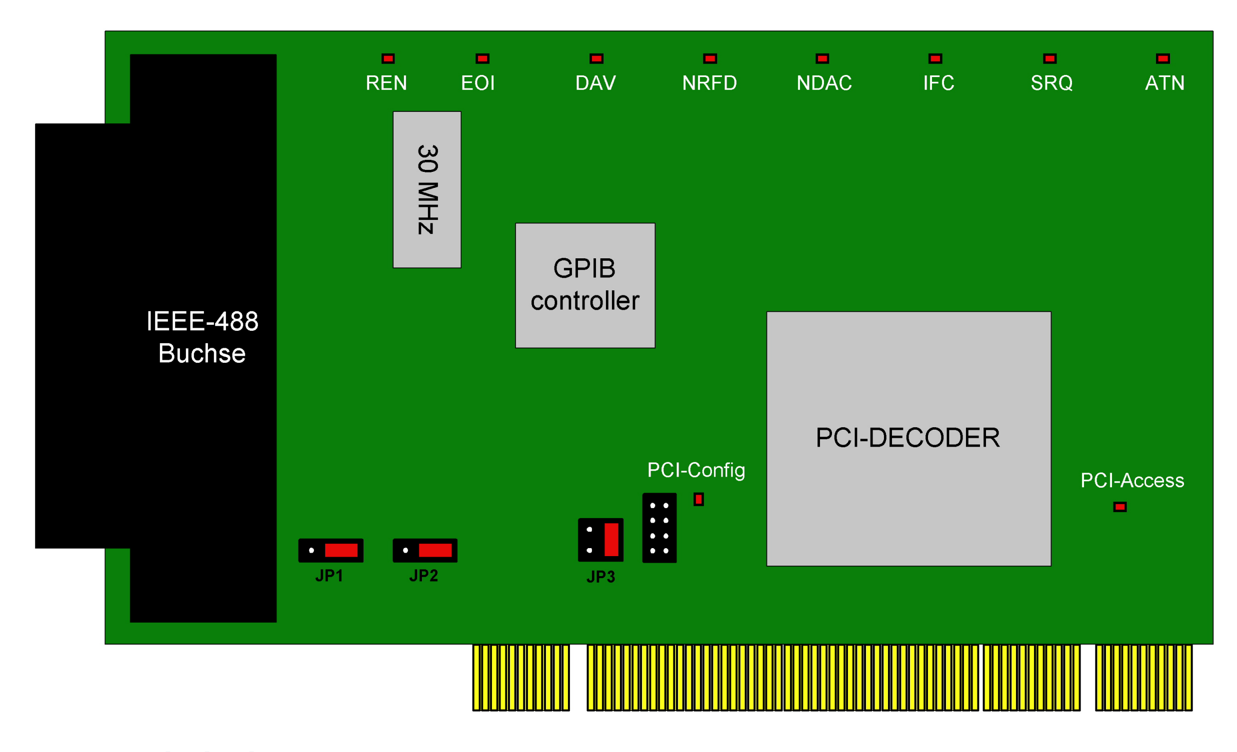
On = I/O-Mode

Off = Memory-Mode



2.4 PCIGPIB-1 (Rev. 3.xx)

2.4.1 Kartenübersicht



2.4.2 Technische Daten

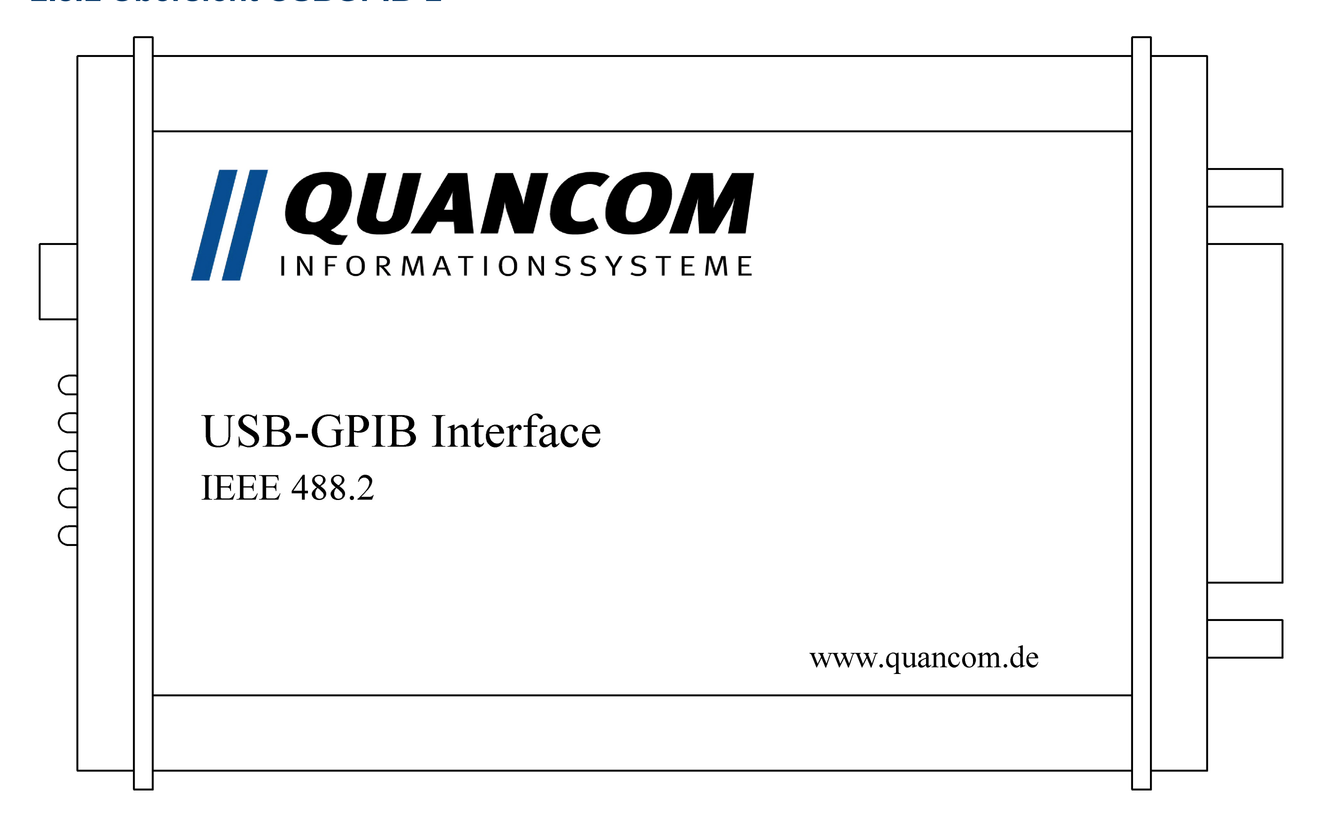
Controller:	µPD7210 kompatibel (mit HW FIFO)
Steckverbinder:	24 pol. IEEE-488 Buchse mit M3.5 Gewinde.
Leuchtdioden:	8 LEDs befinden sich auf der Karte für die Anzeige der Steuerleitungen ATN, SRQ, NDAC, NRFD, DAV, IFC, REN, EOI. Sowie 2 LEDs um einen Zugriff auf die Karte und den PCI-Konfigurations Status anzuzeigen.
Software:	Beispielsoftware in C/C++, Visual Basic, Lotus NotesLabWindows und Delphi

2.4.3 Jumpereinstellungen

+5V Versorgung (JP1)		+3,3V Versorgung (JP2)		Zugriffsmodus (JP3)	
	Von VCC PCI-Bus		Von 3,3V PCI-Bus		I/O-Modus
	Von +5V PCI-Bus		Interne Erzeugung		Memory-Modus

2.5 USBGPIB (Rev. 4.xx)

2.5.1 Übersicht USBGPIB-1

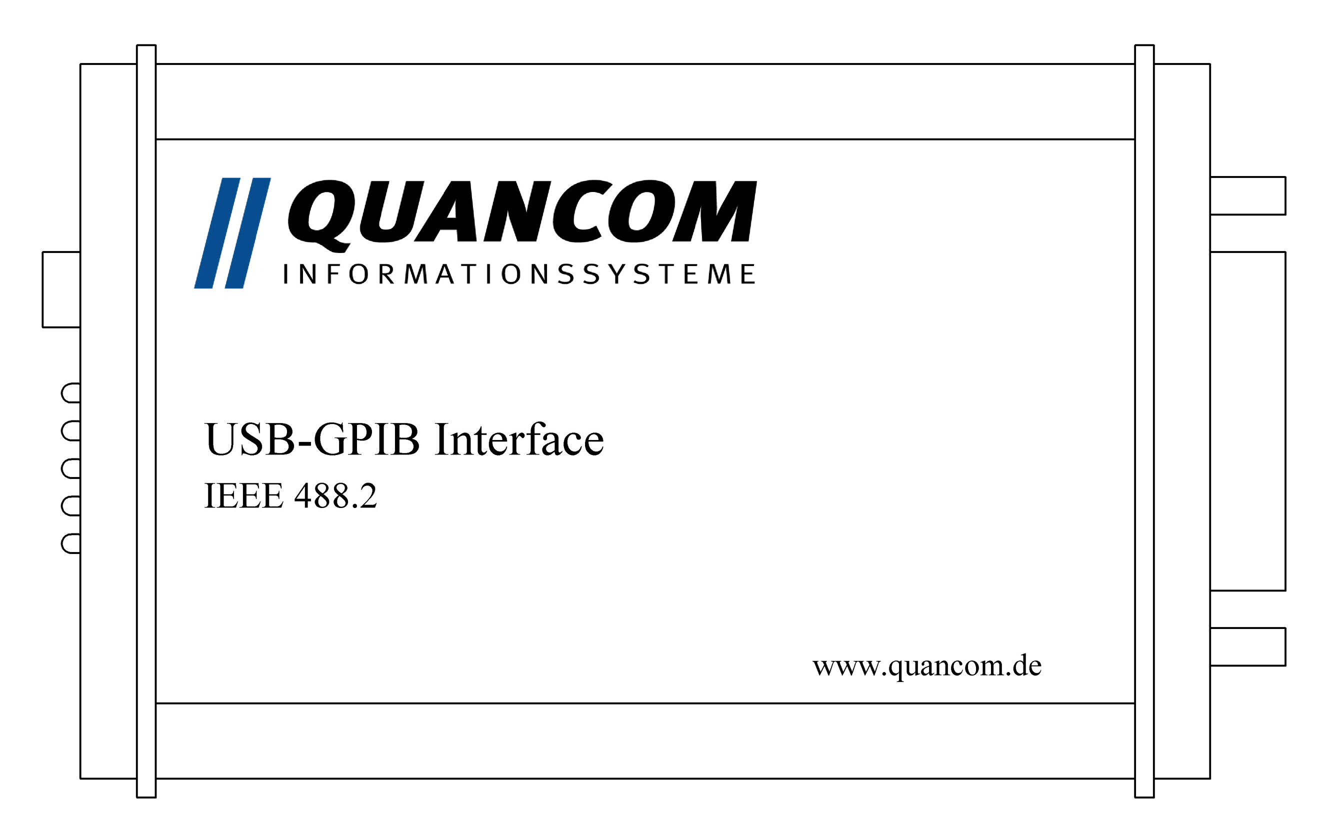


2.5.2 Technische Daten

Controller:	µPD7210 kompatibel (mit HW FIFO)
Steckverbinder:	24 pol. IEEE-488 Buchse mit M3.5 Gewinde
Leuchtdioden:	2 LEDs zeigen den Power und Activation Status an
	4 LEDs für optionale Anwendungen
Software:	Beispielsoftware in C/C++, Visual Basic, Lotus Notes, LabWindows und Delphi

2.6 USBGPIB-1 (Rev. 3.xx)

2.6.1 Abbildung USBGPIB-1



2.6.2 Technische Daten

Controller: µPD7210 kompatibel (mit HW FIFO)

Steckverbinder: 24 pol. IEEE-488 Buchse mit M3.5 Gewinde

Leuchtdioden: 8 LEDs befinden sich an der Seite des Gehäuses für die Anzeige der Steuerleitungen ATN, SRQ, NDAC, NRFD, DAV, IFC, REN, EOI. Sowie 2 LEDs um den USB Status anzuzeigen.

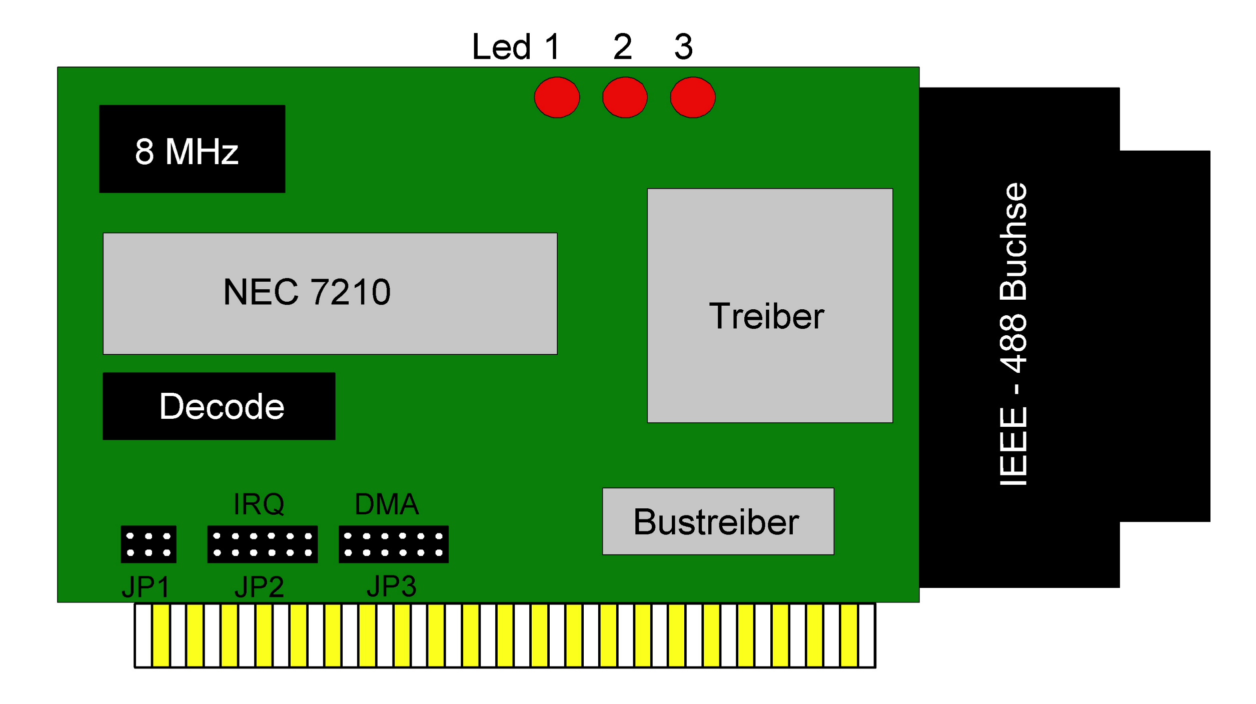
Software: Beispielsoftware in C/C++, Visual Basic, Lotus Notes LabWindows und Delphi

2.7 GPIB-1

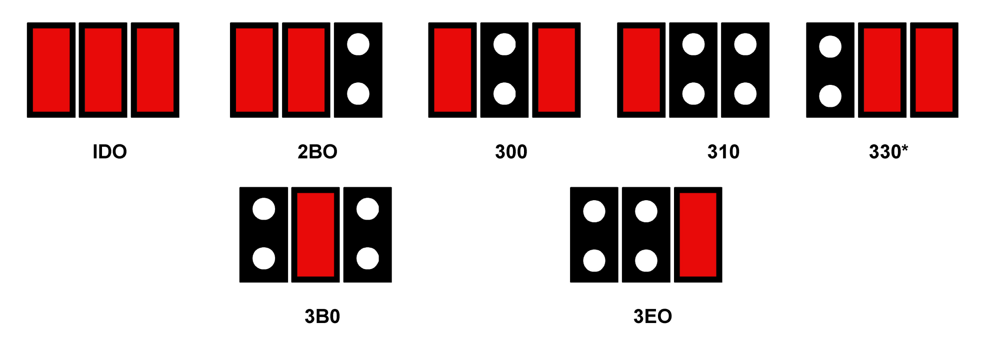
2.7.1 Technische Daten

Controller:	NEC µPD7210
PC-BUS:	Interrupts schaltbar mit: IRQ 2, 3, 4, 5, 6, 7
DMA:	DMA 1,2 oder 3 per Jumper wählbar.
Adreßdekodierung:	sieben verschiedene Adressen per Jumper wählbar: [HEX] 1D0, 2B0, 300, 310, 330, 3B0, 3E0
Datenbus:	8-Bit Datenbus, gepuffert
Steckverbinder:	24 pol. IEEE-488 Buchse mit M3.5 Gewinde
Leuchtdioden:	3 LEDs befinden sich auf der Karte mit Richtungsanzeige der Treiber für Daten- und Kontrolleitungen.
Software:	Beispielsoftware in C, PASCAL und Basic. DEVICE-Treiber für andere Programmiersprachen sowie zur direkten Plotteransteuerung.

2.7.2 Kartenübersicht

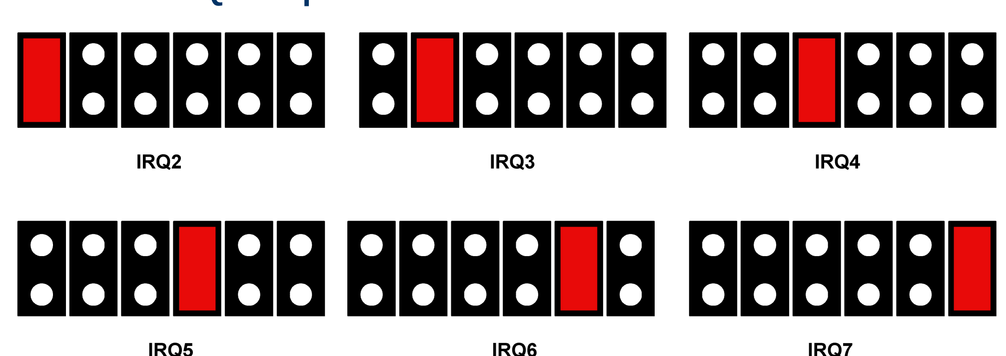


2.8 JP1: Adreßjumper für I/O – Belegung im PC



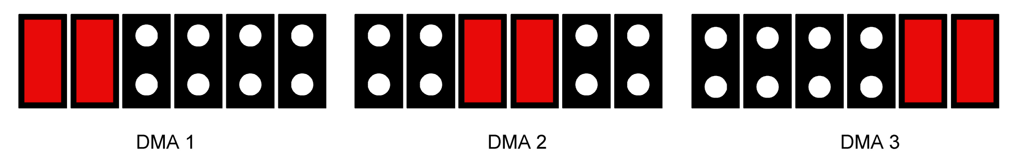
- Standard – Adresse

2.8.1 JP2: IRQ-Jumper



- Standard: Kein IRQ

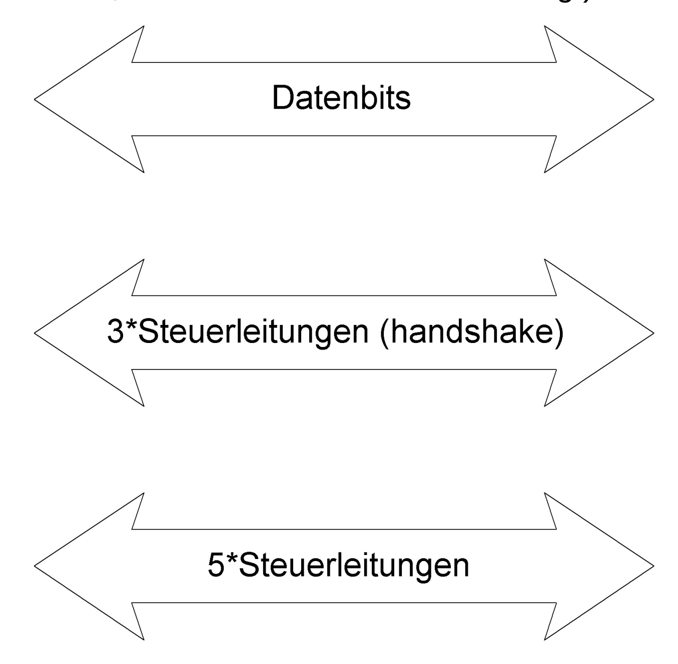
2.8.2 JP3: DMA-Jumper



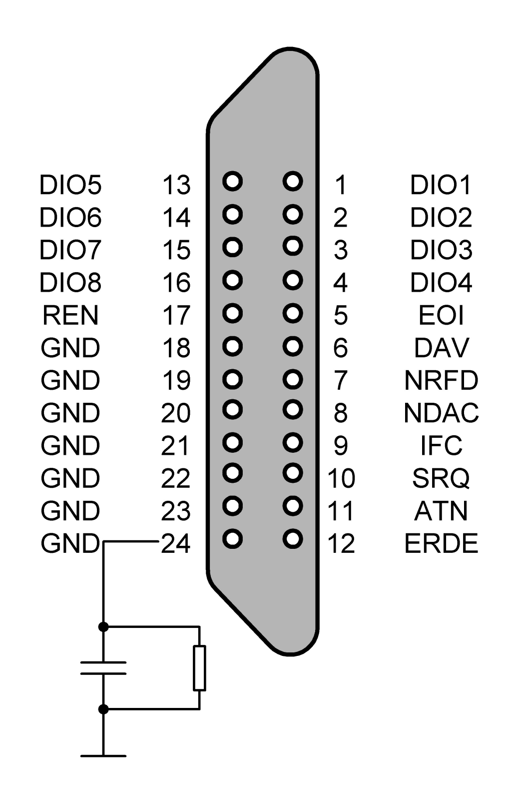
- Standard: Kein DMA

2.8.3 Leuchtdioden

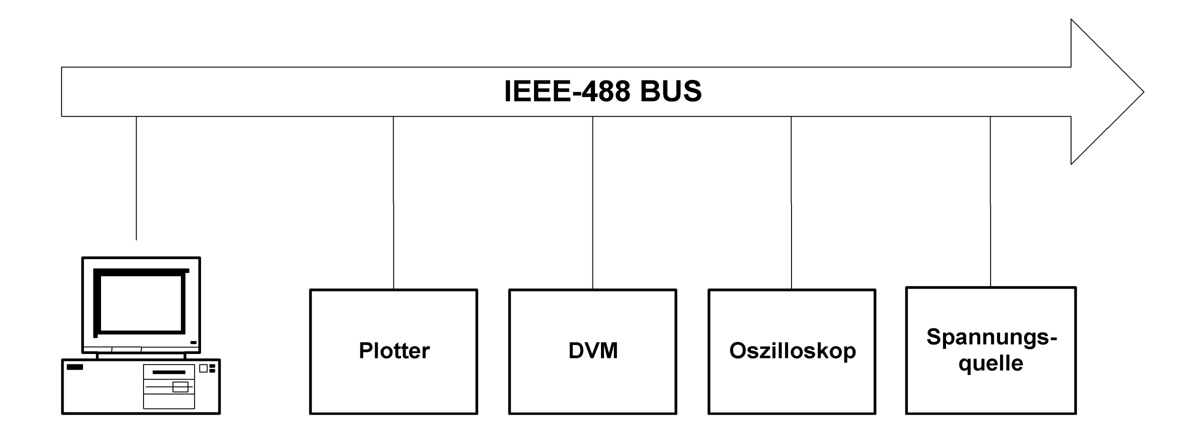
Die drei Leuchdioden (LED 1,2 und 3) signalisieren den Status der folgenden Leitungen: (LED ON - GPIB-Karte sendet; LED OFF - GPIB-Karte emfängt):



2.9 Steckerbelegung USBGPIB1, PCIGPIB1 und GPIB-1



2.10 Konzept



Es können maximal 30 Instrumente an die GPIB-Controllerkarte angeschlossen werden.
Im Beispiel oben sind die Instrumente Plotter, DVM (Digital Voltmeter), Oszilloskop und Spannungsquelle an der Kommunikationsleitung angeschlossen.
Nun kann man z.B. das Voltmeter als 'talker' definieren, die Spannungsquelle als 'listener'.
Möchte man die ankommenden Spannungswerte auch mit dem Oszilloskop analysieren und zusätzlich mit dem Plotter ausgeben, können zusätzlich das Oszilloskop und der Plotter als 'listener' deklariert werden. Nun stehen die vom 'talker' gesendeten Daten an allen drei 'listeners' zur Verfügung.
Die Deklaration des Voltmeters als 'talker' und der anderen Instrumente als 'listener' sowie die Kommunikationsorganisation wird von der GPIB-Controller-Karte übernommen.

3.1 Systemvoraussetzungen

- Personal Computer: Die QUANCOM Karten laufen in einem IBM-AT Computer mit 80X86 oder kompatibel (z.B. 80386 / 80486 / Pentium).
- Bus: Ihr Computer muß den entsprechenden Bus haben (PCI / ISA).



3.2 Sicherheitsanweisungen

Im Interesse Ihrer eigenen Sicherheit und einer einwandfreien Funktion Ihrer neuen QUANCOM-Karte beachten Sie bitte die folgenden Hinweise:

- Ziehen Sie vor dem Öffnen des Rechners den Netzstecker um sicherzustellen, dass das Gerät nicht unter Spannung steht.
- Da PC-Karten empfindlich gegen elektrostatische Aufladungen sind ist es wichtig, sich vorher zu entladen bevor die Karte mit den Händen oder dem Werkzeug berührt wird. Dies geschieht am einfachsten, wenn Sie vorher ein metallisches Gehäuseteil berühren.
- Halten Sie die Karte immer am Rand fest und vermeiden Sie ein Anfassen der IC´s.
- Legen Sie die Karten immer auf einen elektrogeschützten Gegenstand solange sich die Karten noch nicht im PC Gehäuse befinden.



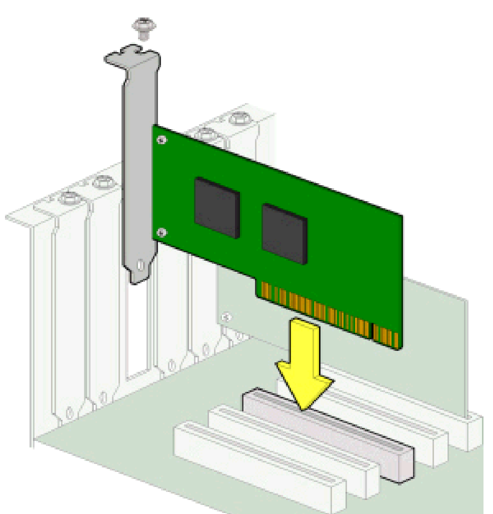
Veränderungen, die ohne ausdrückliche Genehmigung der QUANCOM Informationssysteme GmbH an dem Gerät vorgenommen werden, führen zum Erlöschen der Betriebserlaubnis und der CE Zertifizierung.

3.3 Installation der PCI- / ISAGPIB Karte



Schalten Sie die Stromzufuhr ab und entfernen Sie das Netzkabel bevor Sie eine Karte ein- oder ausbauen. Verhindern Sie immer statische Elektrizität. Schauen Sie dafür bitte unter "Sicherheitsanweisungen" nach.

1. Schalten Sie den Rechner und alle daran angeschlossenen Geräte aus und entfernen Sie die Stromzufuhr.
   Entladen Sie sich vor dem Weiterarbeiten, indem Sie eine Wasserleitung, ein Heizungsrohr oder ein anderes Metallteil mit Erdverbindung berühren.



2. Öffnen Sie den PC. Eventuell müssen Sie einige behindernde Kabel entfernen, merken Sie sich jedoch unbedingt die dazugehörigen Buchsen!

3. Die Einsteckplätze befinden sich am hinteren Ende ihres Rechners. Die Rückwand nicht benutzter Plätze wird von einem Schutzblech verdeckt. Suchen Sie einen freien Einsteckplatz und entfernen Sie das dazugehörige Schutzblech, indem Sie seine Halterungsschraube lösen.

4. Stecken Sie die Erweiterungskarte in den freien Steckplatz. Achten Sie auf festen Sitz und darauf, dass Sie die Karte beim Einstecken senkrecht halten.

5. Befestigen Sie das Halterungsblech der Karte mit der Schraube des Schutzbleches.

6. Schließen Sie das Gehäuse Ihres Rechners und befestigen Sie es mit den Sicherungsschrauben. Kabel, die Sie während des Einbaus gelöst haben, sollten Sie nun wieder einstecken.

7. Stecken Sie die/das Anschlußkabel der Karte in die vorgesehenen Buchsen.

3.4 Installation des USBGPIB-Moduls

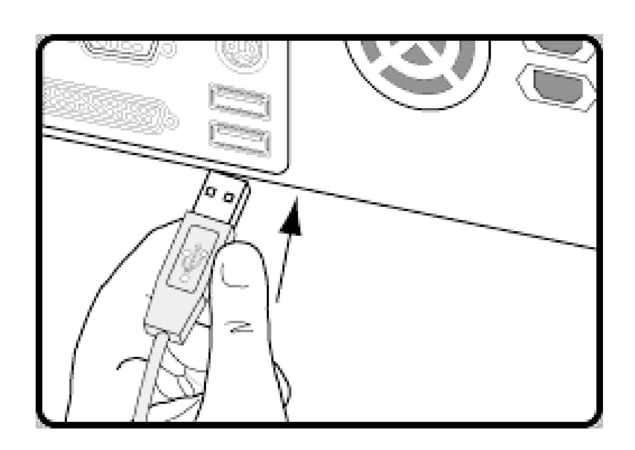
Den PC einschalten, Windows starten und das USB-Verbindungkabel mit dem Gerät und dem USB-Port des PCs verbinden.





Das Modul kann jederzeit während des Betriebs des PCs angeschlossen werden.

Der Ort, an dem sich der USB-Port befindet, ist in der Bedienungsanleitung Ihres PCs beschrieben. Er ist je nach PC-Modell verschieden.



4.1 Welche Software brauche ich ?

Die Software die ich brauche ist abhängig von der jeweiligen Anwendung und Betriebssystem. Um Zugriff per Programm auf die Karte zu bekommen sind bestehen folgende Möglichkeiten:

- **Methode 1**: Direkter I/O Zugriff (Direkter Zugriff auf die Hardware Register der Karte / des Moduls)

- **Methode 2**: High-Level Programmierung (Zugriff auf die Karte über die QLIB) unter Windows XP / 2000 / NT4.x / ME / 98 / 95. Hiermit ist es möglich die Karte, bzw.das Modul z.B. über Visual-C, Visual-Basic, Borland Delphi, Lotus Notes u.a. Compilern und Interpretern anzusprechen.

- **Methode 3**: Installation der QLIB in Zusammenhang mit einem anderen Programm (Beispiele zu dem Einsatz mit Labview von National Instruments finden Sie im Internet unter : **http://www.quancom.de/qprod01/deu/files/download.gpib_vi.zip/$file/gpib_vi.zip** oder nach der Installation der QLIB im Verzeichnis d:\program files\quancom\qlib32\samples\gpib.

Wenn Sie die **Methode 1** und **2** anwenden möchten, benötigen Sie den Quelltext der Anwendung. Sie sind selbst für das Hinzufügen der Befehle in Ihre Anwendung verantwortlich. Um diese Methoden zu benutzen sind Programmier Kenntnisse nötig.

Methode 3 erlaubt einem das man die QUANCOM Karte mit einer bestehenden Software laufen lassen kann z.B. LabView oder HP VEE . Dafür müssen Sie als erstes die QLIB von der Installations CD installieren. Hinweise rund um die QLIB und Installation entnehmen Sie bitte dem QLIB Handbuch, welches sich ebenfalls auf der CD befindet. Auf der CD finden Sie gleichzeitig einige Beispielprogramme für LabView und HP VEE.

4.2 QLIB: High Level Programmierung (Windows XP / 2000 / NT4.x / ME / 98 / 95)

4.2.1 QLIB (QUANCOM Driver Library)

Die QLIB (die Abkürzung für **QUANCOM LIBrary**) bietet die Möglichkeit, alle QUANCOM-Karten unter den Betriebssystemen Windows XP/2000/NT und ME/98/95 und den Programmiersprachen C/C++/Delphi/Visual Basic anzusprechen. Sie wird zu allen QUANCOM-Karten mitgeliefert und gestattet dem Anwender durch die Einfachheit der Befehle, die QLIB in eigene Applikationen einzubinden. Die Befehle und Funktionen gelten für alle Betriebssysteme.

Unterstützte Betriebssysteme:

- Microsoft Windows Vista / XP / 2000 / NT 4.0 / ME / 98 / 95 und Linux

Unterstützte Compiler:

C / C++

- Borland C++ 3.1, 4.x, 5.x
- Microsoft® Visual C++ 1.x, 2.x, 4.x, 5.x, 6.x

Pascal

- Borland Turbo Pascal

Delphi

- Borland Delphi

Basic

- Microsoft® Visual Basic 3.x, 4.x, 5.x; 6.x

Grafische Programmiersprache:

- Agilent VEE von Agilent
- LabView® von National Instruments

4.2.2 Schnellinstallation der QLIB für USB und PCI



Für die Installation der Treiber und Laufzeitumgebung sind Administratorenrechte erforderlich. Ohne die entsprechenden Rechte kann der Treiber und die Laufzeitumgebung nicht korrekt installiert werden.

Windows XP/2000	Windows ME/98
USB-Modul einstecken	USB-Modul einstecken
Rechner starten	Rechner starten
USB-Version wird vom Betriebsystem erkannt, bitte den Pfad zu den Treibern angeben. (Treiber befinden sich im Verzeichnis **WinXP** oder **Win2000** der Installations-CD)	USB-Version wird vom Betriebsystem erkannt, bitte den Pfad zu den Treibern angeben. (Treiber befinden sich im Verzeichnis **WinME** oder **Win98** der Installations-CD)
QLIB, Testprogramme und Beispiele installieren (Im Hauptverzeichnis der Installations-CD **QUANCOM.EXE** starten)	QLIB, Testprogramme und Beispiele installieren (Im Hauptverzeichnis der Installations-CD **QUANCOM.EXE** starten)
Rechner neu starten	Rechner neu starten

4.2.3 Schnellinstallation der Software für ISA und PAR



Für die Installation der Treiber und Laufzeitumgebung sind Administratorenrechte erforderlich. Ohne die entsprechenden Rechte kann der Treiber und die Laufzeitumgebung nicht korrekt installiert werden. Eine detailiertere Installationsanleitung finden Sie in unserem Manual QLIB.

Windows XP/2000/95/98/ME/NT4.x
PAR-Modul anschließen / ISA-Karte einstecken und die Jumpereinstellungen notieren.
Rechner starten
Starten Sie die Datei **QUANCOM.EXE** im Hauptverzeichnis der Installations-CD und folgen Sie den Anweisungen des Programmes.
Rechner neu starten
Öffnen Sie nun die Konfiguration, die sich unter START | Programme | QLIB32 | QLIB 32-Bit Configuration Utility befindet. Klicken Sie hier auf Hinzufügen und wählen nun Ihre Karte aus. Es öffnet sich jetzt automatisch der Dialog mit dem Sie die I/O Adresse der Karte auswählen können. Überprüfen Sie ob die Einstellung der E/A Adresse mit der Jumperstellung auf Ihrer Karte übereinstimmt. Wenn nicht, öffnen Sie bitte durch doppelklicken auf “E/A Basisadresse” den Dialog zur Einstellung E/A Adresse.
Rechner neu starten
PAR-Modul / ISA-Karte mit dem mitgelieferten Testprogramm auf Funktionstüchtigkeit prüfen.

5 QLIB Befehle

Anbei sehen Sie eine auszugsweise Auflistung der QLIB-Befehle für die QUANCOM GPIB Karten. Eine detaillierte Erklärung aller QLIB-Befehle finden Sie bei entsprechender Installation in der QLIB-Windows-Hilfe in Ihrem Startmenu.

5.1 Einfache QLIB Befehle

5.1.1 Verwaltungsfunktionen

QAPINumOfCards

Mit der Funktion **QAPIExtNumOfCards** wird abgefragt, wieviele unterschiedliche Karten- bzw. Modultypen von der QLIB unterstützt werden.

ULONG **QAPINumOfCards** (void);

QAPIGetLastError

Die **QAPIGetLastError** Funktion liefert den letzten Fehlercode des aufrufenden Threads. Der letzte Fehlercode wird dabei für jeden Thread gesondert gespeichert. Mehrere Threads überschreiben ihre Fehlercodes nicht gegenseitig.

ULONG QAPIGetLastError(void);

QAPIGetLastErrorCode

Die **QAPIGetLastErrorCode** Funktion liefert den letzten erweiterten Fehlercode eines vorher aufgerufenen **QAPIGetLastError** Befehls.

ULONG **QAPIGetLastErrorCode**(void);

5.1.2 GPIB - Routinen

QAPIWriteString

Mit der Funktion **QAPIWriteString** wird ein String and das Gerät gesendet. In der aktuellen Fassung der QLIB wird dieser Befehl für die QUANCOM GPIB Controller verwendet. Dieser Befehl ist in der Zukunft auch für RS422/485 Karten vorgesehen.

ULONG **QAPIWriteString** (ULONG **cardid**,ULONG **device**,char* **buffer**,ULONG **chars**, ULONG **mode**);

QAPIReadString

Mit der Funktion **QAPIReadString** wird ein String von einem Gerät empfangen. In der aktuellen Fassung der QLIB wird dieser Befehl für die QUANCOM GPIB Controller verwendet. Dieser Befehl ist in der Zukunft auch für RS422/485 Karten vorgesehen.

ULONG **QAPIReadString** (ULONG **cardid**,ULONG **device**,char* **buffer**,ULONG **chars**, ULONG **mode**);

QAPISpecial

Mit der Funktion **QAPISpecial** wird das Ausführen von kartenspezifischen Funktionen ermöglicht.

ULONG **QAPISpecial** (ULONG **cardid**,ULONG **jobcode**,ULONG **para1**,ULONG **para2**);

5.2 Erweiterte QLIB Befehle

5.2.1 Verwaltungsfunktionen (Erweitert)

QAPIExtOpenCard

Mit der Funktion **QAPIExtOpenCard** wird eine Karte geöffnet.

ULONG **QAPIExtOpenCard** (ULONG **cardid**,ULONG **devnum**);

Funktionsprototypen für Delphi und VB

Parameter

cardid
Gibt die ID der Karte an, welche geöffnet werden soll.

devnum
Gibt die Device-Nummer der Karte an, die geöffnet werden soll.

QAPIExtCloseCard

Mit der Funktion **QAPIExtCloseCard** wird eine Karte geschlossen.

void **QAPIExtCloseCard**(ULONG **cardhandle**);

Funktionsprototypen für Delphi und VB

Parameter

cardhandle
Gibt das Handle einer geöffneten Karte an.

QAPIExtNumOfCards

Mit der Funktion **QAPIExtNumOfCards** wird abgefragt, wieviele unterschiedliche Karten- bzw. Modultypen von der QLIB unterstützt werden.

ULONG **QAPIExtNumOfCards** (void);

Funktionsprototypen für Delphi und VB

Parameter

Diese Funktion benötigt keine Parameter.

QAPIGetLastErrorStringEx

Die **QAPIGetLastErrorStringEx** dient dazu einen String zu erzeugen, der eine lesbare Fehlermeldung aus den QLIB Fehlercodes erzeugt, die von der Funktion **QAPIGetLastError** bzw. **QAPIGetLastErrorCode** zurückgeliefert werden. Ein Beispiel finden Sie unter Fehler codes .

ULONG **QAPIGetLastErrorStringEx**(char* **buffer**, ULONG **buffersize**);

Funktionsprototypen für Delphi und Vb

Parameter

buffer
Zeiger auf einen Puffer der den nullterminierten Fehlerstring erhält.

buffersize
Dieser Parameter gibt die Grösse des übergebenen Puffers in Bytes an

5.2.2 GPIB - Routinen(Erweitert)

QAPIExtWriteString

Mit der Funktion **QAPIWriteString** wird ein String and das Gerät gesendet. In der aktuellen Fassung der QLIB wird dieser Befehl für die QUANCOM GPIB Controller verwendet. Dieser Befehl ist in der Zukunft auch für RS422/485 Karten vorgesehen.

ULONG **QAPIWriteString** (ULONG **cardhandle**,ULONG **device**,char* **buffer**,ULONG **chars**, ULONG **mode**);

Funktionsprototypen für Delphi und Vb

Parameter

cardhandle
Gibt das Handle der Karte der Karte an.

device
Gibt das Gerät auf dem Bus an. Auf dem GPIB Bus ist dieses die Listeneradresse des Empfangsgerätes.

buffer
Zeiger auf den Schreibepuffer der den zu versenden String enthält.

chars
Anzahl der zu versenden Zeichen im Puffer.

mode
Mode zeigt an, wie der Schreibvorgang beendet werden soll.

0	Anhängen von LF an den String und Senden von EOI bei LF.
1	Aktivieren der EOI-Leitung beim letzten Zeichen im String.
2	Anhängen von CR an den String und Senden von EOI bei CR.
3	Anhängen von LF an den String und Senden von EOI bei LF.
4	Anhängen von CR LF an den String und Senden von EOI bei LF.
5	Anhängen von CR an den String, aber kein Senden von EOI.
6	Anhängen von LF an den String, aber kein Senden von EOI.
7	Anhängen von CR LF an den String, aber kein Senden von EOI.
8	Kein Senden von EOI.e

QAPIExtReadString

Mit der Funktion **QAPIReadString** wird ein String von einem Gerät empfangen. In der aktuellen Fassung der QLIB wird dieser Befehl für die QUANCOM GPIB Controller verwendet. Dieser Befehl ist in der Zukunft auch für RS422/485 Karten vorgesehen.

ULONG **QAPIReadString** (ULONG **cardhandle**,ULONG device,**char*** buffer,ULONG **chars**, ULONG **mode**);

Funktionsprototypen für Delphi und Vb

Parameter

cardhandle
Gibt das Handle der Karte der Karte an.

device
Gibt das Gerät auf dem Bus an. Auf dem GPIB Bus ist dieses die Talkeradresse des sendenden Gerätes.

buffer
Zeiger auf den Schreibepuffer der den zu empfangenden String enthalten soll.

chars
Grösse des durch Buffer bereitgestellten Speicher, der für den Empfang von Daten zur Verfügung steht.

mode
Gibt die Endekennung des zu lesenden Strings an:

0 EOS-Zeichen ist LF. Der Lesevorgang wird bei EOI, Anzahl der Bytes oder LF beendet.

1 Kein EOS-Zeichen. Der EOS-Endmodus ist deaktiviert.

2 EOS-Zeichen ist CR. Der Lesevorgang wird bei EOI, Anzahl der Bytes oder CR beendet.

3 EOS-Zeichen ist LF. Der Lesevorgang wird bei EOI, Anzahl der Bytes oder LF beendet.

x Jeder andere Modus bezeichnet die Nummer (dezimal) des gewünschten EOS-Zeichens.

QAPIExtSpecial

Mit der Funktion **QAPISpecial** wird das Ausführen von kartenspezifischen Funktionen ermöglicht.

ULONG **QAPISpecial** (ULONG **cardhandle**,ULONG **jobcode**,ULONG **para1**,ULONG **para2**);

Funktionsprototypen für Delphi und Vb

Parameter

cardhandle
Gibt das Handle der Karte an, die angesprochen werden soll.

jobcode
Gibt die Aktion an, die auf der mit cardid angegebenen Karte ausgelöst werden soll.

para1
Parameter sind abhängig vom jobcode (siehe hierzu auch das Feld Anmerkungen)

para2
Parameter sind abhängig vom jobcode (siehe hierzu auch das Feld Anmerkungen)

Rückgabewert

Wurde die Funktion erfolgreich ausgeführt, so wird ein von jobcode anhängiger Rückgabewert geliefert.

Anmerkung

Für die GPIB Karten GPIB, PCIGPIB und USBGPIB können die folgenden Werte für jobcode angegeben werden:

jobcode	Funktion	para1	para2	Rückgabewert
JOB_READSRQ	Den Status der SRQ Leitung lesen. SRQ wird gesetzt, wenn ein Gerät auf dem Bus Daten im Statusbyte bereit hält. Das ist z.B. bei einem DMM der Fall, wenn das DMM dem PC mitteilen möchte, dass es die Wandlung abgeschlossen hat, und einen Messwert abgeholt werden kann.	---	---	Ungleich 0 wenn ein Gerät die SRQ Leitung aktiviert hat. Sonst 0.
JOB_SERIALPOLL	Führt eine Serial Poll Abfrage auf dem angegebenen Gerät durch. Die Abfrage liefert ein Statusbyte. Der Inhalt des Statusbytes wird durch den Hersteller des jeweiligen Messgerätes festgelegt. Konsultieren Sie hierzu bitte das Manual des jeweiligen Messgerätes.	Talker-adresse	---	Geräteabhän-giges Statusbyte
JOB_GTL	Sendet das Kommando “Goto Local” an das angegebene Gerät.Schaltet das Gerät vom Fernbedienungsmodus in den Modus um, in dem es über das Frontpanel bedient werden kann.	Listen-Adresse	---	Ungleich 0, wenn das Kommando erfolgreich ausgeführt wurde.
JOB_GET	Sendet das Kommando “Group Execute Trigger” an alle Geräte. Das Kommando startet dann auf allen Geräten am Bus eine gerätespezifische Funktion (z.B. Messwert aufnehmen)	---	---	Ungleich 0, wenn das Kommando erfolgreich ausgeführt wurde.
JOB_SDC	Sendet das Kommando “Selected Device Clear” an das angegebene Gerät. Setzt das adressierte Gerät zurück.	Listen-Adresse	---	Ungleich 0, wenn das Kommando erfolgreich ausgeführt wurde.
JOB_LLO	Sendet das Kommando “Local LockOut” an alle Geräte. Schaltet die Möglichkeit aus, die Geräte über das Frontpanel zu bedienen.	---	---	Ungleich 0, wenn das Kommando erfolgreich ausgeführt wurde.
JOB_DCL	Sendet das Kommando “Device Clear” an alle Geräte. Setzt alle Geräte zurück.	---	---	Ungleich 0, wenn das Kommando erfolgreich ausgeführt wurde.
JOB_REN	Aktiviert bzw. deaktiviert die REN Leitung. Wird auf ein Gerät geschrieben geht es danach in den Remote Mode bis die Leitung wieder deaktiviert wird.	= 1 Aktiviert die REN Leitung = 0	---	Ungleich 0, wenn das Kommando erfolgreich ausgeführt wurde.

		Deaktiviert die REN Leitung		
JOB_RESET	Führt einen Hardware Reset auf dem GPIB Controller durch, und überprüft ob die Karte bzw. das Modul vorhanden ist. Da USB Module im Betrieb ausgesteckt werden können liefert die Funktion den Status zurück	---	---	Ungleich 0, wenn das Kommando erfolgreich ausgeführt wurde. = 0 wenn das USB GPIB Modul ausgesteckt wurde, oder die ISA Karte nicht auf der richtigen Portadresse installiert ist. Bei PCI GPIB Karten liefert die Funktion immer einen Wert ungleich 0 zurück.
JOB_TIMEOUT	Setzt das GPIB Session Timeout in Millisekunden. Eine Lese- oder Schreiboperation wird abgebrochen, wenn nach der angegeben Zeit keine weiteren Daten gesendet bzw. empfangen wurden.	Timeout In msec.	---	Ungleich 0, wenn das Kommando erfolgreich ausgeführt wurde.
JOB_READ_TIMEOUT	Liest das GPIB Session Timeout in Millisekunden.	---	---	GPIB Session Timeout
JOB_READ_ DEFAULT_TIMEOUT	Liest das GPIB Default Timeout in Millisekunden. Dabei handelt es sich um den Timeoutwert mit dem eine GPIB Session initialisiert wird, wenn eine GPIB Karte geöffnet wird.	---	---	Ungleich 0, wenn das Kommando erfolgreich ausgeführt wurde.
JOB_WRITE_ DEFAULT_TIMEOUT	Setzt das GPIB Default Timeout in Millisekunden. Dabei handelt es sich um den Timeoutwert mit dem eine GPIB Session initialisiert wird, wenn eine GPIB Karte geöffnet wird.	Timeout In msec.	---	Ungleich 0, wenn das Kommando erfolgreich ausgeführt wurde.

6 GPIB Programmbeispiele

6.1 GPIB Visual-C/C++ Beispiel

Der fettgedruckt Text beschreibt, wie man die QLIB in ein MFC / C++ / C Projekt einbindet. Nach der Installation der QLIB befindet sich das sofort kompilierbare Beispielprojekt im Ordner *d:\program files\quancom\qlib32\samples\gpib\vc* (Gilt nur bei einer Installation in das Standartverzeichnis).

```
// gpib.cpp : Sample project for the GPIB-1, PCIGPIB and USB GPIB Modules
//
// Author: Michael Reimer, QUANCOM Informationssysteme GmbH, Germany
//
// Website: http://www.quancom.de
//
// Information:
//
// To use the QLIB Commands in your source, do the following:
//
// (1) Add statement #include "qlib.h" to your source file.
// (2) Add in the Dialog Menu->Project->Settings->C/C++>Preprocessor
// "$(QLIB_INC)" to the additional include directories entry.
// (3) Add in the Dialog Menu->Project->Settings->Linker->General
// "$(QLIB_LIB)\qlib32.lib" to the additional library and object
// modules directories entry.

#include "stdafx.h"
#include "windows.h"
#include "qlib.h"

int main(int argc, char* argv[])
{
   ULONG result;
   ULONG listener;
   ULONG talker;
   char buffer[1024];

   // The following sequence tries to find the
   // Bus Type ( PCI, ISA or USB ) of the
   // installed GPIB Controller

   ULONG handle = QAPIExtOpenCard(PCIGPIB,0);
   if ( handle == NULL )
      {
         handle = QAPIExtOpenCard(USBGPIB,0);
         if ( handle == NULL )
            {
               handle = QAPIExtOpenCard(GPIB,0);
            }
      }

   // The handle is != NULL if there is a GPIB Controller
    // installed

   if ( handle == NULL )
      {
   MessageBox(NULL, "No QUANCOM GPIB Controller found!", "Error", MB_OK);
         return FALSE;
      }

   // Ok, we found a QUANCOM GPIB Controller Card
   // Now we can send a string to the listner with address 3
   // Change the address to the appropriate address for your
   // device ( normally set by DIP-Switches on the back side )
   // ---------------------------------------------------------
   // PART 1a: Writing a string to the DMM ( The DMM is a the // listener )
   //
```

```
// Listener: A device capable of receiving data over the
        // interface
// when addressed to Listen by the active controller.
       // Examples of such
// devices are printers, programmable power supplies, or any
 // other programmable instrument. There can be up to 14
// Listeners on the GPIB Bus at one time.
// ---------------------------------------------------------
// Select the listener address, which is set by a DIP-Switch
// on the back side of your instrument.

listener = 3;

// Send "z" to the listner 3, which resets the DMM to the
      // initial settings.

char s1[] = "z";
if ( QAPIExtWriteString(handle, listener, (char*)&s1, strlen(s1),0))
   {
      printf("Writing to device %u was successful\n", listener);
   }
else
   {
      printf("Writing to device %u failed\n", listener);
   }

// ---------------------------------------------------------
// PART 1b: Reading a DMM ( The DMM is a talker )
//
// Talker: A device capable of transmitting data over the
// interface when addressed to talk by the active
// controller. Examples of such devices are voltmeters,
// data-acquisition systems, or any other programmable
// instrument. There can be only one addressed talker on the // GPIB Bus at one time.
// ---------------------------------------------------------
// Select the talker address, which is set by a DIP-Switch
// on the back side of your instrument.

talker = 3;

// Read value from DMM with talker address 3

result = QAPIExtReadString(handle, talker, (char*)&buffer, sizeof(buffer), 0);
if (result)
   {
printf("Reading from device %u was: %s\n",talker,   buffer);
   }
else
   {
      printf("Reading from device %u failed\n", talker);
   }



// ---------------------------------------------------------
// PART2: Checking whether a device has requested service
// (SRQ Service Request)
// A device can interrupt the active controller by asserting
// the SRQ line. The SRQ is a single line, and if there are
// multiple devices on the GPIB Bus that have been
// configured to assert an SRQ, the active controller will // have to "poll"
   the devices to figure out which one
    // actually asserted the SRQ.
// More than one device could in principle assert an SRQ at //
   the same time. The active controller can poll the devices
// in one of two ways: serial poll or parallel poll.
//   // ---------------------------------------------------

result = QAPIExtSpecial(handle, JOB_READSRQ, 0, 0);
if (result)
   {
      printf("SRQ was asserted\n");
   }
else
   {
      printf("SRQ was not asserted\n");
   }

// ---------------------------------------------------------
```

```
// PART3: Reading Serial Poll Status from DMM
//
// In a serial poll, the active controller asks each device
// in turn to send back a status byte that indicates whether // the device has asserted the SRQ. Bit 6 of this
// byte (where the bits are numbered 0 through 7) is set
// if the device is requesting service. The definition of
// the other bits is device dependent (under 488.1 at least;
// 488.2 provides a much more concise definition of the
// status byte).
// The program has to perform this same sequence with every
// device it needs to poll.
// -------------------------------------------------------
// We poll all devices from 1 to 15 here. This is normally
// not nescessary and very time consuming. Poll only valid // devices on the GPIB Bus.

ULONG serial_poll_byte = 0;
for ( ULONG device=1;device<15; device++ )
   {
result = QAPIExtSpecial(handle, JOB_SERIALPOLL, device, (ULONG)&serial_poll_byte);
      if ( result )
         {
               if (serial_poll_byte & 0x40)

          // check for bit 6 = device requested service

               {
                printf("Device %u requested service
                    and returned status byte %u\n",
                  device, serial_poll_byte);
            }
         else
            {
                 printf("Device %u no SRQ requested.
                   Status is %u \n", device,
                serial_poll_byte);
            }
         }
      else
         {
            printf("No answer from device %u\n",
              device);
         }
   }

// -------------------------------------------------------
// PART4: Send a command to DMM
//
// The following commands are accepted by all devices on the
// GPIB Bus simultaneously. The address part will be
  // ignored.
//
// - JOB_DCL (Device Clear): The DCL command causes all
//   devices to return to a device dependent initial state.
//
// - JOB_LLO (Local Lockout): The LLO command disables the
        return-to-local front
//   panel key on the device. The user can no longer change
//   the device settings from its front panel.
//
// The following commands need an address and are only
  // accepted by addressed devices. Whether the devices are
// the listeners or the talkers depends on the command. The
 // three commands are as follows:
//
// - JOB_GET (Group Execute Trigger): The GET command tells
 //   all the addressed
//   listeners to perform some device-dependent function,
 //   like take a measurement.
//   GET allows for synchronizing a measurement function
 //   between multiple devices.
//   This is only used in specialized cases.
//
// - JOB_SDC (Selected Device Clear): The SDC command resets
//   the addressed listeners to a device-dependent state. It
 //   performs the same function as DCL,
//   but only resets the addressed listeners, not all the
 //   devices.
//
// - JOB_GTL (Go To Local): The GTL command sets the
```

```
     //   addressed listeners back to
    //   local mode.
    //-----------------------------------------------------------
    // Send DCL to all devices

    result = QAPIExtSpecial(handle, JOB_DCL, 1, NULL);
    if ( !result )
       {
          printf("Command DCL failed\n");
       }

    // Send LLO to all devices

    result = QAPIExtSpecial(handle, JOB_LLO, 1, NULL);
    if ( !result )
       {
          printf("Command LLO failed\n");
       }

    // Send GET to device 3

    result = QAPIExtSpecial(handle, JOB_GET, 3, NULL);
    if ( !result )
       {
          printf("Command GET failed\n");
       }

    // Send SDC to device 3

    result = QAPIExtSpecial(handle, JOB_SDC, 3, NULL);
    if ( !result )
       {
          printf("Command SDC failed\n");
       }

    // Send GTL to device 3

    result = QAPIExtSpecial(handle, JOB_GTL, 3, NULL);
    if ( !result )
       {
          printf("Command GTL failed\n");
       }
    QAPIExtCloseCard(handle);
    return 0;
}
```

6.2 GPIB VB-Beispiel

Der fettgedruckt Text beschreibt, wie man die QLIB in ein Visual-Basic Projekt einbindet. Nach der Installation der QLIB befindet sich das sofort kompilierbare Beispielprojekt im Ordner *d:\program files\quancom\qlib32\samples\gpib\vb* (Gilt nur bei einer Installation in das Standartverzeichnis).

```
' gpib.bas : Sample project for the GPIB-1, PCIGPIB and USB GPIB Modules
'
' Author: Michael Reimer, QUANCOM Informationssysteme GmbH, Germany
'
' Website: http://www.quancom.de
'
' Information:
'
' To use the QLIB Commands in your source, do the following:
' (1) Install the QLIB ( QUANCOM Driver Libarary )
' (2) Add module 'qlib.bas' to your project.
'

Sub Main()

Dim s As String
Dim handle As Long
Dim result As Long
Dim register As Integer
Dim nCardID As Integer
Dim nListener As Long
Dim serial_poll_Byte As Long

' The following sequence tries to find the
' Bus Type ( PCI, ISA or USB ) of the
' installed GPIB Controller

handle = QAPIExtOpenCard(PCIGPIB, 0)

If (handle = 0) Then

     handle = QAPIExtOpenCard(GPIB, 0)

     If (handle = 0) Then

        handle = QAPIExtOpenCard(USBGPIB, 0)

    End If

End If

If (handle = 0) Then
   MsgBox "Unable to find GPIB Controller Card!", 16, "Error"
   Exit Sub
End If

' Ok, we found a QUANCOM GPIB Controller Card
'
' Now we can send a string to the listner with address 3
' Change the address to the appropriate address for your
' device ( normally set by DIP-Switches on the back side )
'
' -------------------------------------------------------------------
' PART 1a: Writing a string to the DMM ( The DMM is a the listener )
'
' Listener: A device capable of receiving data over the interface
' when addressed to Listen by the active controller. Examples of
' such devices are printers, programmable power supplies, or any
' other programmable instrument. There can be up to 14 Listeners on
' the GPIB Bus at one time.
' -------------------------------------------------------------------
'
' Select the listener address, which is set by a DIP-Switch on the
' back side of your instrument.
'

nListener = 3
s = "z"
```

```
' Send 'z' to the listner 3, which resets the DMM to the initial settings.

result = QAPIExtWriteString(handle, nListener, s, Len(s), 0)

If (result) Then
     ' data send to gpib device
Else
     ' failure
     MsgBox "Failure sending data!", vbInformation, "Information"
End If

' ------------------------------------------------------------------
' PART 1b: Reading a DMM ( The DMM is a talker )
'
' Talker: A device capable of transmitting data over the interface
' when addressed to talk by the active controller. Examples of such
' devices are voltmeters, data-acquisition systems, or any other
' programmable instrument. There can be only one addressed talker on
' the GPIB Bus at one time.
' ------------------------------------------------------------------
'
' Select the talker address, which is set by a DIP-Switch on the
' back side of your instrument.
'
nListener = 3
s = Space$(1024) ' create buffer 1024 chars
'
' Read value from DMM with talker address 3

result = QAPIExtReadString(handle, nListener, s, Len(s), 0)

If (result) Then
     ' data successfully read from gpib device

     MsgBox "Read from device " & nListener & " String " & s

Else
     ' failure
     MsgBox "Failure reading data!", vbInformation, "Information"
End If

' ------------------------------------------------------------------
' PART2: Checking whether a device has requested service ( SRQ
' Service Request)
'
' A device can interrupt the active controller by asserting the SRQ
' line. The SRQ is a single line, and if there are multiple devices
' on the GPIB Bus that have been configured to assert an SRQ, the
' active controller will have to 'poll' the devices to figure out
' which one actually asserted the SRQ.More than one device could in
' principle assert an SRQ at the same time. The active controller
' can poll the devices in one of two ways: serial poll or parallel
' poll.
' ------------------------------------------------------------------

result = QAPIExtSpecial(handle, JOB_READSRQ, 0, 0)

If (result = 0) Then
      MsgBox "No service requested", 16, "Information"
Else
      MsgBox "Device has requested service ( SRQ )", 16, "Information"
End If

' ------------------------------------------------------------------
' PART3: Reading Serial Poll Status from DMM
'
' In a serial poll, the active controller asks each device in turn
' to send back a status byte that indicates whether the device has
' asserted the SRQ. Bit 6 of this byte (where the bits are numbered
' 0 through 7) is set if the device is requesting service. The
' definition of the other bits is device dependent (under 488.1 at
' least; 488.2 provides a much more concise definition of the status
' byte).
'
' The program has to perform this same sequence with every device
' it needs to poll.
' ------------------------------------------------------------------
'
' We poll all devices from 1 to 15 here. This is normally not
' nescessary and very time consuming. Poll only valid devices on the
' GPIB Bus.

serial_poll_Byte = 0
```

```
For nListener = 1 To 15

 result = QAPIExtSpecialSP(handle, JOB_SERIALPOLL, nListener, serial_poll_Byte)

   If (result) Then
      ' data successfully read from gpib device
      MsgBox "Serial poll from device " & nListener & " returns = " & Val(serial_poll_Byte)
   Else
      ' failure
      MsgBox "Failure reading data!", vbInformation, "Information"
   End If

Next nListener

' ------------------------------------------------------------------
' PART4: Send a command to DMM
'
' The following commands are accepted by all devices on the GPIB Bus
' simultaneously. The address part will be ignored.
'
' - JOB_DCL (Device Clear): The DCL command causes all devices to
'   return to a device dependent initial state.
'
' - JOB_LLO (Local Lockout): The LLO command disables the return-to-
'   local front panel key on the device. The user can no longer
'   change the device settings from its front panel.
'
' The following commands need an address and are only accepted by
' addressed devices. Whether the devices are the listeners or the
' talkers depends on the command. The three commands are as follows:
'
' - JOB_GET (Group Execute Trigger): The GET command tells all the
'   addressed listeners to perform some device-dependent function,
'   like take a measurement.GET allows for synchronizing a
'   measurement function between multiple devices.This is only used
'   in specialized cases.
'
' - JOB_SDC (Selected Device Clear): The SDC command resets the
'   addressed listeners to a device-dependent state. It performs the
'   same function as DCL,but only resets the addressed listeners,
'   not all the devices.
'
' - JOB_GTL (Go To Local): The GTL command sets the addressed
'   listeners back to local mode.
' ------------------------------------------------------------------

' Send DCL to all devices

result = QAPIExtSpecial(handle, JOB_DCL, 1, 0)

If (Not result) Then
   MsgBox "Command DCL failed!", 16, "Information"
End If

' Send LLO to all devices

result = QAPIExtSpecial(handle, JOB_LLO, 1, 0)

If (Not result) Then
   MsgBox "Command LLO failed!", 16, "Information"
End If

' Send GET to device 3

result = QAPIExtSpecial(handle, JOB_Get, 3, 0)

If (Not result) Then
   MsgBox "Command GET failed!", 16, "Information"
End If

' Send SDC to device 3

result = QAPIExtSpecial(handle, JOB_SDC, 3, 0)

If (Not result) Then
   MsgBox "Command SDC failed!", 16, "Information"
End If

' Send GTL to device 3

result = QAPIExtSpecial(handle, JOB_GTL, 3, 0)
```

```
If (Not result) Then
   MsgBox "Command GTL failed!", 16, "Information"
End If

QAPIExtCloseCard (handle)

End Sub

Private Sub Form_Load()
   Call Main
End Sub
```

6.3 GPIB National Instruments LabView Beispiel

Nach der Installation der QLIB befindet sich die LabView VI's im Ordner *C:\program files\quancom\qlib32\samples\gpib\labview* (Gilt nur bei einer Installation in das Standartverzeichnis).

Die Datei GPIB.VI ist das Beispielprogramm für National Instruments LabView. Um die Kartenfunktionen unter Labview zu testen, können Sie dieses Programm verwenden. Das Beispiel lädt die Dateien gpib_receive.vi und gpib_send.vi, welche die QLIB Funktionen aufrufen.

Die folgenden VI's stehen momentan zur Verfügung:

gpib_receive.vi	Lesen eines Strings von einem Gerät am Bus
gpib_send.vi	Senden eines Strings an ein Gerät am Bus

Die folgenden VI's werden in den nächsten Tagen zur Verfügung gestellt.

gpib_spoll.vi	Serialpoll der Geräte am Bus
gpib_checksrq.vi	Lesen des SRQ-Status auf dem Bus
gpib_command.vi	Senden eines Kommandos auf dem Bus

Bitte überprüfen Sie unsere Website in den nächsten Tagen auf Updates. Updates der QLIB finden Sie unter **http://www.quancom.de/qlib** .

Um die GPIB Kontrollerkarte von LabView aus anzusprechen, sind die folgenden Schritte nötig:

Schritt 0: Installieren Sie die QLIB (QUANCOM Driver Library)
Schritt 1: Fügen Sie die Makros "gpib_receive.vi" und "gpib_send.vi" Ihrem Projekt hinzu.

6.4 GPIB Lotus Notes Beispiel

Der fettgedruckte Text enthält Script Beispiele für die Einbindung der QLIB in Lotus Notes. Das folgende Beispiel stellt nur die Scripte dar, die sich natürlich auch auf Felder in Masken beziehen. Details entnehmen Sie bitte der Beispieldatenbank. Nach der Installation der QLIB befindet sich diese Beispieldatenbank im Ordner *d:\program files\quancom\qlib32\samples\gpib\lotus* (Gilt nur bei einer Installation in das Standartverzeichnis).

Welche Schritte nötig sind, um die QLIB zu eigenen Projekten hinzuzufügen, entnehmen Sie bitte dem Header im Quelltext.

```
' gpib.bas : Sample project for the GPIB-1, PCIGPIB and USB GPIB Modules
'
' Author: Michael Reimer, QUANCOM Informationssysteme GmbH, Germany
'
' Website: http://www.quancom.de
'
' Information:
'
' To use the QLIB Commands in your source, do the following:
' (Step 0: Install the QLIB ( QUANCOM Driver Library )
Step 1: Copy Script Library "qlib" to your database
Step 2: Add Statement  Use "qlib" to section Globals->Options
'

'
' Change the address to the appropriate address for your
' device ( normally set by DIP-Switches on the back side )
'
' ---------------------------------------------------------------
' PART 1: Writing a string to a GPIB / HPIB / IEEE488 DMM ( The DMM
' is a listener ) from Lotus Notes.
' Listener: A device capable of receiving data over the interface
' when addressed to Listen by the active controller. Examples of
' such devices are printers, programmable power supplies, or any
' other programmable instrument. There can be up to 14 Listeners on
' the GPIB Bus at one time.
' ---------------------------------------------------------------

Sub SendString()

   Dim ws As New NotesUIWorkspace
   Dim uidoc As NotesUiDocument

   Dim nDevice As Long
   Dim nListener As Long
   Dim s As String
   Dim handle As Long
   Dim result As Long
   Dim nCardID As Integer
   Dim CardName As String
   Dim Listener As String

   Set uidoc =  ws.CurrentDocument

   CardName =  uidoc.FieldGetText("SelectedCard")

   If ( CardName = "") Then
      Msgbox "Please select the GPIB Card!" ,16,"Error"
      Exit Sub
   End If

' QLIB supports up to 8 GPIB Controller

   nDevice = Val(Right$(CardName, 1))

   If (Instr(CardName, "PCIGPIB")) Then
      nCardID = PCIGPIB
   Elseif (Instr(CardName, "USBGPIB")) Then
      nCardID = USBGPIB
   Else
      nCardID = GPIB
   End If
```

```
' Retrieve Listener Address

   Listener =  uidoc.FieldGetText("Listener")

   nListener = Val(Right$(Listener, 2))

' Retrieve Text to send to the GPIB Device

   s = uidoc.FieldGetText("StringToSend")

   handle = QAPIExtOpenCard(nCardID, nDevice)

   If (handle <> 0) Then

      result = QAPIExtWriteString(handle, nListener, s, Len(s), 0)

      If (result) Then
      ' data send to gpib device
      Else
      ' failure
         Msgbox "Failure sending data!", 16, "Information"
      End If

   End If


End Sub

' ------------------------------------------------------------------
' PART 2: Reading a GPIB / IEEE-488 / HPIB Device from Lotus Notes
'
' Talker: A device capable of transmitting data over the interface
' when addressed to talk by the active controller. Examples of such
' devices are voltmeters, data-acquisition systems, or any other
' programmable instrument. There can be only one addressed talker on
' the GPIB Bus at one time.
' ------------------------------------------------------------------
'
' Select the talker address, which is set by a DIP-Switch on the
' back side of your instrument.
'

Sub ReadString()

   Dim ws As New NotesUIWorkspace
   Dim uidoc As NotesUiDocument

   Dim nDevice As Long
   Dim nListener As Long
   Dim s As String
   Dim handle As Long
   Dim result As Long
   Dim nCardID As Integer

   Dim CardName As String
   Dim Listener As String

   Set uidoc =  ws.CurrentDocument

   CardName =  uidoc.FieldGetText("SelectedCard")

   If ( CardName = "") Then
      Msgbox "Please select the GPIB Card!" ,16,"Error"
      Exit Sub
   End If

' QLIB supports up to 8 GPIB Controller

   nDevice = Val(Right$(CardName, 1))

   If (Instr(CardName, "PCIGPIB")) Then
      nCardID = PCIGPIB
   Elseif (Instr(CardName, "USBGPIB")) Then
      nCardID = USBGPIB
   Else
      nCardID = GPIB
   End If

' Retrieve Listener Address
```

```
   Listener =  uidoc.FieldGetText("Talker")

   nListener = Val(Right$(Listener, 2))

' Create string buffer

   s = Space$(256)

   handle = QAPIExtOpenCard(nCardID, nDevice)

   If (handle <> 0) Then

      result = QAPIExtReadString(handle, nListener, s, Len(s), 0)

      If (result) Then
      ' data successfully read from gpib device

         Call uidoc.FieldSetText("String", s)

      Else
      ' failure
         Msgbox "Failure reading data!", vbInformation, "Information"
      End If

   End If

End Sub


' ------------------------------------------------------------------
' PART3: Checking whether a device has requested service ( SRQ
' Service Request)
'
' A device can interrupt the active controller by asserting the SRQ
' line. The SRQ is a single line, and if there are multiple devices
' on the GPIB Bus that have been configured to assert an SRQ, the
' active controller will have to "poll" the devices to figure out
' which one actually asserted the SRQ.More than one device could in
' principle assert an SRQ at the same time. The active controller
' can poll the devices in one of two ways: serial poll or parallel
' poll.
' ------------------------------------------------------------------

Sub ReadSRQ()

   Dim ws As New NotesUIWorkspace
   Dim uidoc As NotesUiDocument

   Dim nDevice As Long
   Dim nListener As Long
   Dim s As String
   Dim handle As Long
   Dim result As Long
   Dim nCmd As Long
   Dim nCardID As Integer
   Dim CardName As String
   Dim Listener As String

   Set uidoc =  ws.CurrentDocument

   CardName =  uidoc.FieldGetText("SelectedCard")

   If ( CardName = "") Then
      Msgbox "Please select the GPIB Card!" ,16,"Error"
      Exit Sub
   End If

' QLIB supports up to 8 GPIB Controller

   nDevice = Val(Right$(CardName, 1))

   If (Instr(CardName, "PCIGPIB")) Then
      nCardID = PCIGPIB
   Elseif (Instr(CardName, "USBGPIB")) Then
      nCardID = USBGPIB
   Else
      nCardID = GPIB
   End If

   handle = QAPIExtOpenCard(nCardID, nDevice)
```

```
   If (handle <> 0) Then

      result = QAPIExtSpecial(handle, JOB_READSRQ, 0, 0)

      If (result = 0) Then
         Msgbox "No service requested",16,"Info"
      Else
         Msgbox "Device has requested service ( SRQ )",16,"Info"
      End If

   End If

End Sub

' ------------------------------------------------------------------
' PART4: Reading Serial Poll Status from DMM
'
' In a serial poll, the active controller asks each device in turn
' to send back a status byte that indicates whether the device has
' asserted the SRQ. Bit 6 of this byte (where the bits are numbered
' 0 through 7) is set if the device is requesting service. The
' definition of the other bits is device dependent (under 488.1 at
' least; 488.2 provides a much more concise definition of the status
' byte).
'
' The program has to perform this same sequence with every device
' it needs to poll.
' ------------------------------------------------------------------
'
Sub SerialPoll()
   Dim ws As New NotesUIWorkspace
   Dim uidoc As NotesUiDocument

   Dim nDevice As Long
   Dim nListener As Long
   Dim s As String
   Dim handle As Long
   Dim result As Long
   Dim pollbyte As Long
   Dim nCardID As Integer
   Dim CardName As String
   Dim Listener As String

   Set uidoc =  ws.CurrentDocument

   CardName =  uidoc.FieldGetText("SelectedCard")

   If ( CardName = "") Then
      Msgbox "Please select the GPIB Card!" ,16,"Error"
      Exit Sub
   End If

' QLIB supports up to 8 GPIB Controller

   nDevice = Val(Right$(CardName, 1))

   If (Instr(CardName, "PCIGPIB")) Then
      nCardID = PCIGPIB
   Elseif (Instr(CardName, "USBGPIB")) Then
      nCardID = USBGPIB
   Else
      nCardID = GPIB
   End If

' Retrieve Listener Address

   Listener =  uidoc.FieldGetText("Listener2")

   nListener = Val(Right$(Listener, 2))

' Create string buffer

   s = Space$(256)

   handle = QAPIExtOpenCard(nCardID, nDevice)

   If (handle <> 0) Then

      result = QAPIExtSpecialSP(handle, JOB_SERIALPOLL, nListener, pollbyte)

      If (result) Then
```

```
      ' data successfully read from gpib device

         Call uidoc.FieldSetText("String2", "Hex: " & Hex(Val(pollbyte)) & " Dec: " & (Val(pollbyte)))

      Else
      ' failure
         Msgbox "Failure reading data!", vbInformation, "Information"
      End If

   End If
End Sub


' ------------------------------------------------------------------
' PART5: Send a command to DMM
'
' The following commands are accepted by all devices on the GPIB Bus
' simultaneously. The address part will be ignored.
'
' - JOB_DCL (Device Clear): The DCL command causes all devices to
'   return to a device dependent initial state.
'
' - JOB_LLO (Local Lockout): The LLO command disables the return-to-
'   local front panel key on the device. The user can no longer
'   change the device settings from its front panel.
'
' The following commands need an address and are only accepted by
' addressed devices. Whether the devices are the listeners or the
' talkers depends on the command. The three commands are as follows:
'
' - JOB_GET (Group Execute Trigger): The GET command tells all the
'   addressed listeners to perform some device-dependent function,
'   like take a measurement.GET allows for synchronizing a
'   measurement function between multiple devices.This is only used
'   in specialized cases.
'
' - JOB_SDC (Selected Device Clear): The SDC command resets the
'   addressed listeners to a device-dependent state. It performs the
'   same function as DCL,but only resets the addressed listeners,
'   not all the devices.
'
' - JOB_GTL (Go To Local): The GTL command sets the addressed
'   listeners back to local mode.
' ------------------------------------------------------------------

Sub SendCommand()

   Dim ws As New NotesUIWorkspace
   Dim uidoc As NotesUiDocument

   Dim nDevice As Long
   Dim nListener As Long
   Dim s As String
   Dim handle As Long
   Dim result As Long
   Dim nCmd As Long
   Dim nCardID As Integer
   Dim CardName As String
   Dim Listener As String
   Dim CommandName As String

   Set uidoc =  ws.CurrentDocument

   CardName =  uidoc.FieldGetText("SelectedCard")
   CommandName =  uidoc.FieldGetText("CommandList")

   If ( CardName = "") Then
      Msgbox "Please select the GPIB Card!" ,16,"Error"
      Exit Sub
   End If

' QLIB supports up to 8 GPIB Controller

   nDevice = Val(Right$(CardName, 1))

   If (Instr(CardName, "PCIGPIB")) Then
      nCardID = PCIGPIB
   Elseif (Instr(CardName, "USBGPIB")) Then
      nCardID = USBGPIB
   Else
      nCardID = GPIB
```

```
   End If

' Retrieve Listener Address

   Listener =  uidoc.FieldGetText("Listener3")

   nListener = Val(Right$(Listener, 2))

   handle = QAPIExtOpenCard(nCardID, nDevice)

   If (handle <> 0) Then

      Select Case Left$(   CommandName,3)
      Case "GTL": nCmd = JOB_GTL
      Case "SDC": nCmd = JOB_SDC
      Case "GET": nCmd = JOB_GET
      Case "LLO": nCmd = JOB_LLO
      Case "DCL": nCmd = JOB_DCL
      End Select

      result = QAPIExtSpecial(handle, nCmd, nListener, 0)

      If (result) Then
    ' cmd successfully send to gpib device

      Else
      ' failure
         Msgbox "Failure sending command!", vbInformation, "Information"
      End If

   End If
End Sub
```

6.5 GPIB Borland Delphi Beispiel

Der fettgedruckt Text beschreibt, wie man die QLIB in ein Delphi Projekt einbindet. Nach der Installation der QLIB befindet sich das sofort kompilierbare Beispielprojekt gpib.pas im Ordner *d:\program files\quancom\qlib32\samples\gpib\delphi* (Gilt nur bei einer Installation in das Standartverzeichnis).

Welche Schritte nötig sind, um die QLIB zu eigenen Projekten hinzuzufügen, entnehmen Sie bitte dem Header im Quelltext.

```
program GPIB_CONSOLE_APP;

{*
// gpib.pas : Sample project for Borland Delphi shows how to
// program the GPIB-1, PCIGPIB and USB GPIB Modules
//
// Author: Michael Reimer, QUANCOM Informationssysteme GmbH, Germany
//
// Website: http://www.quancom.de
// Product:
// GPIB PCI Controller http://www.quancom.de/qprod01/deu/pb/pcigpib_1.htm
// GPIB ISA Controller http://www.quancom.de/qprod01/deu/pb/GPIB_1.htm
// GPIB USB Controller http://www.quancom.de/qprod01/deu/pb/usb_gpib_1.htm
// Information:
//
// To use the QLIB Commands in your source, do the following:
//
// (1) Add statement #include "qlib.pas" to your source file.
// (2) Copy QLIB.PAS from QLIB Installation Directory
//     d:\program files\quancom\qlib32\include to your
//     working directory
*}

{$APPTYPE CONSOLE}

{$INCLUDE QLIB.pas}

{$X+}

var handle: longint;
    result: longint;
    listener: longint;
    talker: longint;
    buffer: string;
    s: string;
    serial_poll_byte: longint;
    device: longint;

begin

{*
// The following sequence tries to find the
// Bus Type ( PCI, ISA or USB ) of the
// installed GPIB Controller
*}

serial_poll_byte := 3;

result := LongInt(@serial_poll_byte);

result := result + 1;

writeln(result);

handle := QAPIExtOpenCard(PCIGPIB,0);
if ( handle = 0 ) then
   begin
   handle := QAPIExtOpenCard(USBGPIB,0);
   if ( handle = 0 ) then
      begin
      handle := QAPIExtOpenCard(GPIB,0);
      end;
   end;

{*
```

```
// The handle is <> NULL if there is a GPIB Controller installed
*}

if ( handle = 0 ) then
   begin
       s := 'No QUANCOM GPIB Controller found!';
       writeln(s);
       halt(0);
   end;

{*
// Ok, we found a QUANCOM GPIB Controller Card

// Now we can send a string to the listner with address 3
// Change the address to the appropriate address for your
// device ( normally set by DIP-Switches on the back side )

// ---------------------------------------------------------------------------
// PART 1a: Writing a string to a IEEE 488, GPIB, HPIB Device
//
// Listener: A device capable of receiving data over the interface
// when addressed to Listen by the active controller. Examples of such
// devices are printers, programmable power supplies, or any other
// programmable instrument. There can be up to 14 Listeners on the GPIB Bus
// at one time.
// ---------------------------------------------------------------------------

// Select the listener address, which is set by a DIP-Switch on the
// back side of your instrument.
*}

listener := 3;

// Send "z" to the listener 3, which resets the DMM to the initial settings.

s := 'z'#0;

if ( QAPIExtWriteString(handle, listener, Pchar(s), Length(s),0) = 0) then
   begin
      writeln('Writing to device ', listener, ' was successful.');
   end
else
   begin
        writeln('Writing to device ', listener, ' failed.');
   end;

{*
// ---------------------------------------------------------------------------
// PART 1b: Reading a DMM or any other IEEE-488 device
//
// Talker: A device capable of transmitting data over the interface when
// addressed to talk by the active controller. Examples of such devices
// are voltmeters, data-acquisition systems, or any other programmable
// instrument. There can be only one addressed talker on the GPIB Bus at one
// time.
// ---------------------------------------------------------------------------

// Select the talker address, which is set by a DIP-Switch on the
// back side of your instrument.
*}

talker := 3;
SetLength(buffer,1024);

{*
// Read value from DMM with talker address 3
*}

result := QAPIExtReadString(handle, talker, PChar(buffer), Length(buffer), 0);

if ( result <> 0 ) then
   begin
      writeln('Reading from device', talker, ' was ', buffer);
   end
else
    begin
      writeln('Reading from device ', talker, ' failed.');
    end;

{*
// ---------------------------------------------------------------------------
// PART2: Checking whether a device has requested service ( SRQ Service Request)
//
```

```
// A device can interrupt the active controller by asserting the SRQ line. The
// SRQ is a single line, and if there are multiple devices on the GPIB Bus that
// have been configured to assert an SRQ, the active controller will have
// to "poll" the devices to figure out which one actually asserted the SRQ.
// More than one device could in principle assert an SRQ at the same time. The
// active controller can poll the devices in one of two ways: serial poll
// or parallel poll.
// ---------------------------------------------------------------------------
*}

result := QAPIExtSpecial(handle, JOB_READSRQ, 0, 0);

if (result <> 0) then
   begin
        writeln('SRQ was asserted.');
   end
else
    begin
         writeln('SRQ was not asserted.');
    end;

{*
// ---------------------------------------------------------------------------
// PART3: Reading Serial Poll Status from DMM
//
// In a serial poll, the active controller asks each device in turn to
// send back a status byte that indicates whether the device has asserted
// the SRQ. Bit 6 of this byte (where the bits are numbered 0 through 7) is set
// if the device is requesting service. The definition of the other bits
// is device dependent (under 488.1 at least; 488.2 provides a much more
// concise definition of the status byte).
//
// The program has to perform this same sequence with every device
// it needs to poll.
// ---------------------------------------------------------------------------

// We poll all devices from 1 to 15 here. This is normally not nescessary and
// very time consuming. Poll only valid devices on the GPIB Bus.
*}

serial_poll_byte := 0;

for device := 1 to 15 do
    begin

         result := QAPIExtSpecial(handle, JOB_SERIALPOLL, device, LongInt(@serial_poll_byte));

         if ( result <> 0 ) then
            begin
                 if ((serial_poll_byte and $40) = 0) then
                    begin
                         {* check for bit 6 = device requested service *}
                         writeln('Device ', device, ' requested service and returned status byte ' , serial_poll_byt
                    end
                 else
                     begin
                          writeln('Device ', device, ' no SRQ requested. Status is ', serial_poll_byte);
                     end;
            end
         else
            begin
                 writeln('No answer from device ', device);
            end;
    end;

{*
// ---------------------------------------------------------------------------
// PART4: Send a command to DMM
//
// The following commands are accepted by all devices on the GPIB Bus
// simultaneously. The address part will be ignored.
//
// - JOB_DCL (Device Clear): The DCL command causes all devices to return to a device
// dependent initial state.
//
// - JOB_LLO (Local Lockout): The LLO command disables the return-to-local front
// panel key on the device. The user can no longer change the device settings
// from its front panel.
//
// The following commands need an address and are only accepted by addressed
// devices. Whether the devices are the listeners or the talkers depends on the
// command. The three commands are as follows:
//
```

```
// - JOB_GET (Group Execute Trigger): The GET command tells all the addressed
// listeners to perform some device-dependent function, like take a measurement.
// GET allows for synchronizing a measurement function between multiple devices.
// This is only used in specialized cases.
//
// - JOB_SDC (Selected Device Clear): The SDC command resets the addressed listeners
// to a device-dependent state. It performs the same function as DCL,
// but only resets the addressed listeners, not all the devices.
//
// - JOB_GTL (Go To Local): The GTL command sets the addressed listeners back to
// local mode.
// ------------------------------------------------------------------------

// Send DCL to all devices

*}

result := QAPIExtSpecial(handle, JOB_DCL, 1, NULL);

if ( result = 0 ) then
   begin
        writeln('Command DCL failed\n');
   end;

{* Send LLO to all devices *}

result := QAPIExtSpecial(handle, JOB_LLO, 1, NULL);

if ( result = 0 ) then
   begin
        writeln('Command LLO failed\n');
   end;

{* Send GET to device 3 *}

result := QAPIExtSpecial(handle, JOB_GET, 3, NULL);

if ( result = 0 ) then
   begin
        writeln('Command GET failed\n');
   end;


{* Send SDC to device 3 *}

result := QAPIExtSpecial(handle, JOB_SDC, 3, NULL);

if ( result = 0 ) then
   begin
        writeln('Command SDC failed\n');
   end;


{* Send GTL to device 3 *}

result := QAPIExtSpecial(handle, JOB_GTL, 3, NULL);

if ( result = 0 ) then
   begin
        writeln('Command GTL failed\n');
   end;

QAPIExtCloseCard(handle);

end.
```

6.6 Das GPIB Test-Programm

Über das Menü **Start | Programme | QLIB 32-Bit | Program | PCIGPIB, ISA GPIB-1** and **USBGPIB** Controller können Sie das Testprogramm starten, mit dem die Karte getestet werden kann. Dieses Tool befindet sich nur im Menü, wenn auch die Beispielprogramme mitinstalliert wurden. Den Quelltext für dieses Beispielprogramm, finden Sie im Ordner "*d:\program files\quancom\qlib32\samples\gpib\vb*" .

6.7 Funktionsprototypen für Delphi und Vb Parameter

ULONG (ULONG cardhandle,ULONG device,char* buffer,ULONG chars,ULONG mode);

cardhandle:

Gibt das Handle einer geöffneten Karte an.

device:

Gibt das Gerät auf dem Bus an.Auf dem GPIB Bus ist dieses die Talkeradresse des sendenden Gerätes.

buffer:

Zeiger auf dem Schreibepuffer, der den zu empfangenden String enthalten soll.

chars:

Grösse des durch den Buffer bereitgestellten Speicher, der für den Empfang von Daten zur Verfügung steht.

mode:

Gibt die Endekennung des zu lesenden Strings an:

0 EOS-Zeichen ist LF. Der Lesevorgang wird bei EOI, Anzahl der Bytes oder LF beendet

1 Kein EOS-Zeichen. Der EOS-Endmodus ist deaktiviert

2 EOS-Zeichen ist CR. Der Lesevorgang wird bei EOI, Anzahl der Bytes oder CR beendet

3 EOS-Zeichen ist LF. Der Lesevorgang wird bei EOI, Anzahl der Bytes oder LF beendet.x

Jeder andere Modus bezeichnet die Nummer (dezimal) des gewünschten EOS-Zeichens.

Rückgabewert:

Wurde die Funktion erfolgreich ausgeführt, so wird eine Anzahl der empfangenen Zeichen größer Null geliefert. Im durch den Buffer übergebenen Speicherblocks befindet sich die Antwort des Gerätetes. Wenn die Funktion fehlgeschlagen ist, so ist der Rückgabewert immer 0L.

Anmerkungen: Siehe auchQAPIWriteString, QAPIReadString, QAPIExtWriteString, QAPIExtWriteStringEx[...]

7 GPIB-1 Device-Treiber (MS DOS)

7.1 Installation

Der mitgelieferte Device-Treiber "GPIB_DRV.SYS" wird benötigt, um die GPIB-Controller-Karte als Gerät ansteuern zu können.
Um diese Option zu nutzen, müssen Sie den Treiber auf Ihrem Rechner installieren:

1. Legen Sie die mitgelieferte Programm-CD in Laufwerk D: ein.

2. Geben Sie an der MS-DOS-Eingabeaufforderung ein:
   "COPY D:\PRODUKTE\GPIB\GPIB1\DOS_BSP\DEVDRV\GPIB_DRV.SYS C:\" Sollten Sie Probleme haben, so schlagen Sie bitte im MS-DOS Benutzerhandbuch nach.

3. Nun muß der Treiber aktiviert werden. Dazu müssen Sie in der CONFIG.SYS die Zeile DEVICE=C:\GPIB_DRV.SYS ergänzen. Sie können dies unter Verwendung des MS-DOS-Editors tun. Geben Sie dazu "EDIT C:\CONFIG.SYS" ein.

4. Bewegen Sie mit den Pfeiltasten den Cursor an das Ende der Datei und fügen Sie ein: "DEVICE=C:\GPIB_DRV.SYS"

5. Wählen Sie im Dateimenü den Befehl "Speichern" und anschließend den Befehl "Ende".

6. Um den Treiber einsetzen zu können, müssen Sie nun Ihren Rechner neu starten.

7.1.1 Kommandoparameter

Für die weitere Steuerung des Device-Treibers stehen Ihnen die folgenden Parameter zur Verfügung:

/pxxx Übergibt die Portadresse xxx (200..FFF) im Hexadezimalformat.
(Standard: 330 HEX)

/yxx Legt das Zeichen mit der Nummer xx (00..FF) zum Einleiten der Kommandoeingabe fest. **(Voreingestellt ist das Semikolon ";")**. Wird dieses Zeichen direkt als erstes Zeichen nach dem Öffnen der Datei GPIB gesendet, so werden die nachfolgenden Zeichen als Kommando interpretiert. Die Kommandosequenz wird durch Senden eines Semikolons beendet.

Beispiel: Nach dem Öffnen der Datei GPIB zur Ausgabe senden Sie ";/d0D;" und anschließend den Text, der an die Adresse 0D verschickt werden soll (Kommando /dxx siehe weiter unten)

/tx Setzt die Zeitspanne x (0..9) in Sekunden fest, die bis zu einem Timeout-Fehler verstreichen muß. **(Standard: 1)**

/ux Ermöglicht oder verhindert Abbruch der Übertragung mit einem Tastendruck (0: Abbruch nicht möglich, 1: Abbruch durch den Benutzer erlaubt, **Standard:1**)

Hinweis: Ein Abbruch durch den Benutzer führt immer zu einem "allgemeinen Fehler"!

/ex Legt die Endekennung fest, nach der die übergebenen Daten versandt oder im internen Puffer gespeichert werden. Gültige Werte für x:

1: CR+EOI

2: CR (Standard)

3: LF+EOI

4: LF

5: CR+LF+EOI

6: CR+LF

7: LF+CR

8: EOI (nur empfangen)

/sxx Gibt die Talkeradresse xx (1..1E) an.

/dxx Gibt die Listeneradresse xx (1..1E) an.

7.1.2 Beispiele

DEVICE = C:\GPIB_DRV.SYS p330

Der Treiber wird unter Beibehaltung sämtlicher Voreinstellungen geladen, das heißt:
Die Portadresse ist 330 hex,
das Kommandozeichen ist das Semikolon,
die Timeout-Zeit beträgt eine Sekunde,
Abbruch ist nicht möglich,
die Endekennung ist CR.

DEVICE = C:\GPIB_DRV.SYS /p300 /e4 /t2 /u0 /y07

Die Portadresse wird festgelegt auf 300 hex.
Als Endkennung fungiert das Zeichen "LF",
die Timeoutzeit wird auf zwei Sekunden festgesetzt.
Ein Unterbrechen der Datenübertragung durch einen Tastendruck ist nicht möglich;
das Zeichen, welches die Kommandoübergabe einleitet ist 07 hex, also die Glocke (Bell).

7.2 Programmierbeispiel in BASIC:

Auch während des Betriebs können die Parameter der Karte verändert werden. Hierzu dient das oben beschriebene Kommandozeichen. Seine Voreinstellung ist das Semikolon. Um nun die Parameter zu verändern, muß direkt nach dem Öffnen als erstes Zeichen das Kommandozeichen übertragen werden. Nun folgen die Parameter, die geändert werden sollen, anschließend ein ; (Semikolon) und dann die zu übertragenden Daten.

```
10 OPEN "GPIB" FOR OUTPUT AS #1
20 PRINT #1,";/d0e; T0V1T1"
30 CLOSE #1
```

Zeile 10 weist der GPIB-Controllerkarte den Ausgabestream 1 zu. Sie kann nun wie eine Datei angesprochen werden. Der Ausgabebefehl in Zeile 20 adressiert nun zunächst das Gerät mit der Adresse 0e als Listener und gibt T0V1T1 als Daten aus. Der Befehl PRINT gibt im Anschluß an die Daten automatisch noch ein CR aus, das als voreingestellte Endekennung die Übertragung der Daten auslöst. In Zeile 30 wird der Ausgabestream 1 geschlossen.

Auch das Einlesen von Daten ist über den Device-Treiber möglich:

```
10 OPEN "GPIB" FOR INPUT AS #1
20 INPUT #1,A$
30 PRINT A$
40 CLOSE #1
```

Zunächst wird in Zeile 10 der GPIB-Controllerkarte der Eingabestream 1 zugewiesen. In Zeile 20 werden nun die anstehenden Daten eingelesen und in Zeile 30 auf dem Bildschirm ausgegeben. In Zeile 40 wird der Eingabestream 1 geschlossen.

7.3 Programmierbeispiele in C:

```
/* Senden mit der GPIB-Karte */

#include <stdio.h>
#include <fcntl.h>

void main(void)
{
int handle;
char string[20];
unsigned length;

handle=open("GPIB",O_WRONLY);        /* öffnet GPIB zum Schreiben und übergibt die Adresse an handle */
strcpy(string,";/d0e;T0V1T1");               /* Kopiert den Ausgabestring in eine Variable */
length=strlen(string);        /* Die Anzahl der zu übertragenden Zeichen bestimmen */
write(handle,string,length);         /* Schreibt den Ausgabestring nach handle */
close(handle);                               /* Schließt handle */
}
```

```
/* Empfangen mit der GPIB-Karte */

#include <stdio.h>
#include <fcntl.h>

void main(void)
{
int handle;
char string[255];

handle=open("GPIB",O_RDONLY); /* öffnet GPIB zum Lesen und übergibt die Adresse an handle */
read(handle,string,255);             /* Liest Daten von der GPIB-Karte
                                             (max. 255 Zeichen   */printf("%s",string);
                                             Gibt die eingelesenen Daten auf dem                        Bil
close(handle);                       /* Schließt handle */
}
```

7.4 Programmierbeispiele in PASCAL:

```
program Senden_mit_GPIB;

uses crt;

var handle : text;

begin
  assign(handle,'GPIB');                    { Ordnet GPIB der Dateivariablen handle zu }
  reset(handle);                                 { Öffnet GPIB }
  writeln(handle,';/d0e;T0V1T1');     { Gibt die Daten aus }
  close(handle);                                 { Schließt GPIB }
end.
```

```
program Empfangen_mit_GPIB;

uses crt;

var handle     : text;
eingabe : string;

begin
  assign(handle,'GPIB');                    { Weist GPIB der Dateivariablen handle zu }
  reset(handle);                                 { Öffnet handle }
  readln(handle,eingabe);                   { Liest Daten von GPIB }
  writeln(eingabe);                         { Gibt die Daten auf dem Bildschirm aus }
  close(handle);                                 { Schließt GPIB }
end.
```

8.1 Kunden Support und Hilfe



Sie benötigen Hilfe?

Wenn Sie nicht wissen was Sie während einer Installation tun müssen, oder wie die Karte in Betrieb genommen wird, lesen Sie bitte dieses Handbuch.

! Tip !

Im Kapitel ”Frequently asked questions” (Häufig gestellte Fragen) sind einige Antworten auf häufig gestellte Fragen. Sie können Ihnen bei der Problemlösung behilflich sein. Auf der QUANCOM Installations CD finden Sie in Textform die Datei README.TXT, welche alle wichtigen Änderungen beinhaltet.

! Wichtig !

Wenn Sie weitere Fragen haben, kontaktieren Sie unser Support-Team. Für diesen Fall halten Sie bitte folgende Informationen bereit:

- Genauer Karten-Typ
- Version der Treiber
- Version der QLIB
- Betriebssystem, Hardware-Ausstattung und Bussystem
- Name und Version von dem Programm, welches den Fehler ausgibt
- Eine genaue Fehlerbeschreibung (versuchen Sie den Fehler zu wiederholen, um diesen besser beschreiben zu können)

Wen kann ich erreichten?

Die QUANCOM Internet Webseite
www.quancom.de

Per Fax
+49 22 36 / 89 92 - 49

Per E-Mail:
support@quancom.de

Adresse:
QUANCOM INFORMATIONSSYSTEME GmbH
In der Flecht 14
50389 Wesseling

Wenn Sie Hilfe brauchen, erreichen Sie uns unter:
QUANCOM Hotline Deutschland
0 22 36 / 89 92 - 20

Montags - Donnerstag
von 9:00 bis 18:00
Freitags
von 9:00 bis 17:00

Aktuelle Treiber

Auf unserer Internetseite http://www.quancom.de können sie immer die neusten Treiber Versionen und Updates finden. Zudem finden Sie ebenfalls viele andere Informationen und die ”Frequently asked questions (FAQ’s)”. Bevor Sie uns kontaktieren, überprüfen Sie ob die neueste Version der QUANCOM Software installiert ist.

Reparatur

Wenn Sie nicht genau wissen, ob die QUANCOM Karte defekt ist, rufen Sie unsere QUANCOM Hotline an:

Tel.: **+49 22 36 / 89 92 – 20**

Bevor Sie uns die Karte zur Reparatur schicken, rufen Sie unsere Hotline an:

Tel.: **+49 22 36 / 89 92 – 20**

Wenn Sie uns die Karte zurückschicken, legen Sie diese bitte in die Originalverpackung oder eine adäquate Verpackung, um einen Transportschaden zu verhindern. Zusätzlich bitten wir Sie, uns eine Kopie der Originalrechnung mitzuschicken.

8.2 Technisches Support Formular

Wenn Sie einen Internetzugang haben, öffnen Sie folgende URL in Ihrem Browser: **http://www.quancom.de/quancom/qshop.nsf/techniksupport?OpenForm&deu**

Füllen Sie das Formular komplett aus bevor Sie sich an QUANCOM Informationssysteme GmbH wenden. Wenn Sie andere QUANCOM Hardware oder Software nutzen, fügen Sie das bitte dem Formular hinzu.

Name: ________________________________

Firma: ________________________________

Adresse: ________________________________

Telefon: ________________________________

Fax: ________________________________

Computer / Prozessor: ________________________________

Betriebssystem: ________________________________

Grafikkarte: ________________________________

Maus: ________________________________

QUANCOM Karte ________________________________

Andere installierte Karten: ________________________________

Festplatte (Kapazität, frei): ________________________________

Das Problem ist: ________________________________

Auflistung der Fehlermeldung: ________________________________

____

____

Folgende Schritte führen zur Wiederholung des Problems:

____

____

____

____

____

8.3 Hardware und Software Konfigurationsformular

Dieses Formular hilft Ihnen die Einstellungen der Hardware und Software aufzulisten. Füllen Sie das Formular komplett aus bevor Sie sich an QUANCOM Informationssysteme GmbH wenden und nutzen Sie das Formular ebenfalls um die aktuelle Konfiguration nachzuschlagen.

- **QUANCOM Produkt:**

Name / Name der Karte ______________________

Interrupt Level ______________________

DMA Kanal ______________________

Basis I/O Adresse ______________________

Betriebssystem ______________________

- **Andere Informationen**

Computer Model ______________________

Prozessor ______________________

Taktfrequenz ______________________

Grafikkarte ______________________

Betriebssystem ______________________

Programmiersprache ______________________

Programmiersprachen-Version ______________________

- **Andere Karten im System**

Basis I/O-Adresse anderer Karten ______________________

DMA Kanäle anderer Karten ______________________

Interrupt Level anderer Karten ______________________

8.4 Dokumentations Formular

QUANCOM Informationssysteme GmbH möchte Ihren Kommentar zum Produkt und zu der über diese Dokumentation oder eines Produktes. Diese Informationen helfen uns unsere Qualität zu verbessern.

Titel: USB-GPIB-1 / PCIGPIB-1 / GPIB-1

Erstellungsdatum: 25.11.2011

Nehmen Sie Stellung zur Kompetenz, Übersichtlichkeit und Inhalt dieses Handbuches. Wenn Sie Fehler im Handbuch entdecken notieren Sie sich bitte die Seitenzahl.

Vielen Dank für Ihre Hilfe.

Name: ____

Firma: ____

Adresse: ____

Telefon: ____

Fax: ____

Kommentar: ____

____

____

____

____

Email an: **support@quancom.de**

Fax an: **+49 2236 89 92 49**

Adresse: **QUANCOM Informationssysteme GmbH**
In der Flecht 14
50389 Wesseling

8.5 Warenzeichen

Linux ist ein eingetragenes Warenzeichen von Linus Torvalds.
MS, MS-DOS, Microsoft, Visual Basic, Windows, Windows Vista/XP/2000/NT/ME/98/95 sind eingetragene Warenzeichen von Microsoft Corporation.
XT und PS/2 sind Warenzeichen und IBM, OS/2 und AT sind eingetragene Warenzeichen der International Business Machines Corporation.
Intel, Pentium ist ein eingetragenes Warenzeichen von Intel Corporation.
USB ist ein eingetragenes Warenzeichen von USB Implementers Forum Inc.
JAVA ist ein eingetragenes Warenzeichen von Sun Microsystems.
DELPHI und Pascal sind ein eingetragene Warenzeichen von Borland Corporation.
PCI ist ein eingetragenes Warenzeichen von PCI Special Interest Group.
PCI Express ist ein eingetragenes Warenzeichen der PCI-SIG.
Nationalinstruments, LABVIEW ist ein eingetragenes Warenzeichen von Nationalinstruments Corporation.
Agilent VEE ist ein eingetragenes Warenzeichen von Agilent Technologies.
Ethernet ist ein eingetragenes Warenzeichen der Xerox Corporation.

Bei anderen Produkt- und Firmennamen, die in dieser Anleitung erwähnt werden, könnte es sich um Marken ihrer jeweiligen Eigentümer handeln.